UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

INVESTIGAÇÃO E DESENVOLVIMENTO DE UM SISTEMA AUTOMÁTICO DE DETECÇÃO, MONITORIZAÇÃO E ANÁLISE DA PROPAGAÇÃO DE WORMS EM REDES EMPRESARIAIS

Tiago Manuel Simões Sequeira

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2011

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

INVESTIGAÇÃO E DESENVOLVIMENTO DE UM SISTEMA AUTOMÁTICO DE DETECÇÃO, MONITORIZAÇÃO E ANÁLISE DA PROPAGAÇÃO DE WORMS EM REDES EMPRESARIAIS

Tiago Manuel Simões Sequeira

Projecto orientado pelo Prof. Doutor António Casimiro Ferreira da Costa
e co-orientado pelo Eng. José António dos Santos Alegria

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2011

Agradecimentos

Sem algumas pessoas creio que poderia chegar até onde já cheguei mas certamente nada seria igual, poderia ter passado melhores ou piores momentos mas nunca iguais ou tão memoráveis.

Nem sempre a minha caminhada foi fácil, pois venho de famílias humildes mas muito orgulhosas, que não voltam as costas a um desafio. Este foi o espírito que meus pais me incutiram desde muito novo e que, muito sinceramente, eu agradeço imenso.

Agradeço ainda o facto de meus pais me terem dado uma educação de excelência para a vida e não menos importante, todo o apoio financeiro ao longo da minha licenciatura e mestrado, esta acção foi crucial para que todo o meu percurso académico se desenrolasse da melhor forma possível.

Especificamente agradeço ao meu pai Fernando por ter aquele orgulho tão saudável, por me ter ajudado sempre que precisei e mesmo quando não precisei esteve lá, por ser o meu amigo de sempre.

À minha mãe Luisa agradeço por ser a mulher que é, por ter sempre uma palavra de conforto para me dar, por me transmitir aquela paz e ao mesmo tempo aquela intensa força afectiva que me fez ver sempre a luz ao fundo do túnel através da inteligência e do trabalho.

No leque familiar estão ainda incluídos os meus avós que estiveram sempre a meu lado apoiando-me em qualquer situação.

Agradeço aos meus amigos e colegas do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa que trabalharam comigo em projectos durante a minha licenciatura e mestrado em Engenharia Informática.

Agradeço ainda ao professor António Casimiro pela ajuda prestada e pela constante motivação, ao Eng. José Alegria por ter acreditado em mim e por ter tido a audácia de propor um projecto tão ambicioso e por fim agradeço ao Eng. Pedro Inácio pela disponibilidade no acompanhamento deste projecto.

Aos meus pais por "tudo".
Ao meu irmão.

Resumo

Neste projecto pretende-se alertar em teoria e na prática a problemática da segurança de informação em grandes redes empresariais. Estamos a falar de infra-estruturas críticas, por exemplo, no sério mercado das telecomunicações onde existe uma grande preocupação em relação a ataques ou intrusões vindas do exterior em detrimento da preocupação em relação a ataques que tenham origem na própria rede interna.

Para protecção de uma rede empresarial tipicamente são utilizados mecanismos tradicionais como *firewalls* e/ou *Intrusion Detection Systems (IDS)*, no entanto deve-se ter em atenção o facto destas técnicas serem eficientes na protecção contra utilizadores não autorizados e exteriores à rede empresarial, mas muitas vezes ineficientes no que diz respeito a utilizadores autorizados que possam ter já sido infectados por *malware*. Nestas situações, muito dificilmente existirá a garantia de uma rede empresarial segura contra a propagação de *malware* que tenha origem nas máquinas dos utilizadores autorizados (*i.e.* estagiários, colaboradores, etc).

Este projecto surge na tentativa de corrigir esta potencial lacuna de segurança em grandes redes empresariais. Neste contexto é apresentada uma solução completa, o *Worm Monitoring System (WMS)*, que permite a captura automática de tráfego supostamente malicioso que tenha origem na rede empresarial. Para tal, foram criados componentes de *software* que tratam automaticamente das importantes tarefas de identificação, análise dinâmica e alarmística de *malware* propagado internamente (*worms* e *bots*) que possa estar envolvido no tráfego capturado por uma federação de sondas (*honeypots*) estrategicamente distribuídas pela rede. Num ambiente empresarial é requerido ter soluções não intrusivas, assim como soluções leves, eficientes, de fácil integração e acima de tudo produtivas, pelo que houve especial preocupação na idealização e construção de uma arquitectura distribuída para o *WMS*.

Para validar a solução implementada assim como outras aplicações da solução, foi realizada numa fase final, uma avaliação experimental completa do *WMS*, a partir do *Worm Monitoring System interface (WMSi)* que permitiu gerar resultados estatísticos e informação acerca das tendências deste tipo de ataques.

Palavras-chave: Segurança, *Honeypot/Honeynet*, *Worm/Bot*, Monitorização

Abstract

This project is an attempt to correct a potentially dangerous security gap in large enterprise networks, in this context a solution is presented, the WMS, which allows automatic capture of alleged malicious traffic that originates on the corporate network, including mechanisms of back-end, as the mwmonitor prototype, which automatically handles the important task of identifying and dynamic analyzing of internally malware spread like worms and bots that may be involved in captured traffic by the strategically distributed probes on the corporate network.

In a business environment are required to have non-intrusive solutions, as well as lightweight solutions, efficient, easy integration and above all productive, and there was particular concern in the design and construction a decentralized architecture for the WMS well as the choice of constituent technologies.

As a result, after the creation of security metrics, the system also allows the monitoring (WMSi) protection status of a large corporate network with regard to the occurrence of internal propagation of malware.

To validate the implemented solution as well as other applications of the solution was performed in a final phase, an experimental evaluation in which they extract some interesting statistical results and information about attack trends.

Keywords: Security, *Honeypot/Honeynet*, *Worm/Bot*, Monitoring

Conteúdo

Lista de Figuras **xvi**

Lista de Tabelas **xix**

1 Introdução **1**
1.1 Motivação 2
1.2 Objectivos 3
1.2.1 Metodologia de Investigação 4
1.2.2 Tarefas 4
1.3 Empresa e equipa de trabalho 6
1.4 Estrutura do documento 6

2 Sistema *Pulso* **9**
2.1 Objectivos 9
2.2 Arquitectura 10
2.3 Resumo 11

3 Captura, Identificação e Análise de *malware* **13**
3.1 *Malware* 13
3.1.1 Objectivo 14
3.1.2 Vírus 14
3.1.3 *Trojan Horse* 14
3.1.4 *Spyware* 14
3.1.5 *Worm* 15
3.1.6 *Bot/Botnet* 20
3.1.7 Técnicas de evasão de *software* anti-*malware* 21
3.2 Captura de *malware* 21
3.2.1 *Honeypots* 21
3.3 Identificação/detecção e análise de *malware* 24
3.3.1 Identificação/detecção de *malware* 25
3.3.2 Análise de *malware* 26
3.4 Resumo 27

4 Plataformas utilizadas **29**
4.1 *Dionaea* . . . 29
4.1.1 Características . . . 29
4.1.2 Arquitectura . . . 30
4.1.3 Configuração . . . 30
4.1.4 Conectividade . . . 31
4.1.5 Protocolos . . . 31
4.1.6 Funcionamento . . . 32
4.2 *VirusTotal* . . . 35
4.3 *Anubis* . . . 37
4.3.1 Arquitectura e componentes . . . 38
4.3.2 Ferramentas utilizadas . . . 39
4.3.3 Resultados de análise dinâmica . . . 40
4.4 Projectos relacionados . . . 40
4.4.1 *Leurre.com* . . . 40
4.4.2 *WOMBAT* . . . 41
4.5 Resumo . . . 41

5 *WMS* **43**
5.1 Aproximação . . . 43
5.2 Arquitectura . . . 44
5.3 Componentes internos . . . 45
5.3.1 *receivehttp* . . . 45
5.3.2 *mwmonitor* . . . 48
5.3.3 Mecanismo de *log* . . . 56
5.3.4 *WMSi* . . . 57
5.4 Metodologia . . . 59
5.4.1 Tecnologias usadas . . . 59
5.4.2 Recursos disponíveis . . . 60
5.5 Resumo . . . 60

6 Avaliação experimental **63**
6.1 Ambiente e condições . . . 63
6.2 Procedimento . . . 64
6.3 Cenários de utilização . . . 64
6.3.1 Cenário 1 . . . 65
6.3.2 Cenário 2 . . . 75
6.3.3 Cenário 3 . . . 81
6.3.4 Cenário 4 . . . 86
6.4 Resumo . . . 87

7 Discussão **89**
7.1 Trabalho preliminar 89
7.2 Abordagens alternativas 89
7.2.1 *dionaea* e *mwcollectd* 89
7.2.2 *Remote sandbox service* e *Local sandbox service* 90
7.3 Funcionamento interno 92
7.4 Recursos 93
7.5 Resultados 93

8 Conclusão **95**
8.1 Divulgação 96
8.2 Trabalho futuro 96

Bibliografia **103**

Lista de Figuras

1.1 *Action Research* 4

2.1 Arquitectura de *collectors*, *targets* e *loggers* 11

3.1 Fases de um ataque automatizado 16
3.2 *Conficker* 19

4.1 *dionaea* 31
4.2 *Bind shell payload* 33
4.3 *Reverse shell payload* 33
4.4 *Exec payload* 34
4.5 *VirusTotal* 36
4.6 *VirusTotal Web Report* 36
4.7 *Anubis Web Report* 37
4.8 *Anubis* 38

5.1 *WMS* 45
5.2 *receivehttp* 46
5.3 *mwmonitor* 49
5.4 *mailrobot* 53
5.5 *artifactsrobot* 55
5.6 *WMSi* 57

6.1 *WMSi* - página de entrada 76
6.2 Ocorrências de ataque por hora no mês de Junho 77
6.3 Fontes de ataque no mês de Junho 77
6.4 *Malware* identificado/detectado e analisado no mês de Junho 78
6.5 Serviços atacados no mês de Junho 78
6.6 Ocorrências de ataque por hora no mês de Julho 79
6.7 Fontes de ataque no mês de Julho 80
6.8 *Malware* identificado/detectado e analisado no mês de Julho 80
6.9 Serviços atacados no mês de Julho 81
6.10 *Attack Topology Schema* com os últimos 5 ataques no mês de Julho . 82

6.11 Sumário da análise dinâmica do *Net-Worm.Win32.Kido.ih* 82
6.12 Sumário da análise dinâmica do *Net-Worm.Win32.Kolab.acem* 83
6.13 Sumário da análise dinâmica do *Trojan.Win32.Midgare.ayfl* 83
6.14 Últimos 7 dias de ocorrências de ataque 84
6.15 Ocorrências de ataque por mês 84

Lista de Tabelas

3.1 Tabela comparativa dos níveis de actividade de *honeypots* 23

4.1 *Fingerprint* do evento de ataque enviada a partir da sonda para o servidor central. 35

5.1 *Fingerprint* completa para alarmística. 52
5.2 *VT* e *Anubis*: tecnologias usadas. 59
5.3 *WMS*: tecnologias usadas. 60
5.4 *mailrobot* e *artifactsrobot*: tecnologias usadas. 60

6.1 Alterações no *Log Level* do *dionaea*. 67
6.2 *p0f*: parâmetros de execução. 71
6.3 Medição do *real execution time* para o componente *mwmonitor* 74
6.4 Medição do *real execution time* para o componente *mailbinaryrobot* . 86
6.5 Medição do *real execution time* para o componente *artifactsrobot* . . . 87

7.1 *VirusTotal*, *NoVirusThanks* e *Jotti*: comparação. 91

Capítulo 1

Introdução

O aumento no número de *malware* nas empresas cria uma série de desafios e riscos para os profissionais de segurança. Os *worms* e os *bots*, em particular o *Zeus, Conficker, Kolab, Limbo, Torpig/Mebroot/Sinowal* e *SilentBanker*, permitem capturar na surdina uma grande variedade de dados e credenciais de utilizadores *on-line*, inclusive informações essenciais aos negócios.

Como a distribuição de *malware* tem sido, aparentemente, direccionada aos consumidores *on-line*, várias organizações ainda desconhecem os riscos em potencial apresentados por recursos infectados na empresa. As organizações devem compreender até que ponto o *malware* começa a comprometer as informações relacionadas à empresa para que as políticas e os controlos possam ser ajustados adequadamente a fim de prevenir a perda de dados.

A complexidade organizacional, humana e tecnológica que está inerente às grandes empresas aumenta fortemente os riscos destas de tornarem um alvo bastante propício a ataques informáticos. O facto de existir um nível de heterogeneidade humana bastante elevado numa grande empresa, dificulta a monitorização da utilização de serviços e o respectivo comportamento indevido por parte dos utilizadores internos e utilizadores externos. As empresas baseiam-se muitas vezes no bom senso dos seus utilizadores, o que poderá não ser uma boa prática.

Uma das grandes preocupações que coloca uma série de desafios aos profissionais de segurança de informação numa empresa, reside na possibilidade da ocorrência de *malware* propagado automaticamente pela rede interna através das máquinas ou dispositivos de utilizadores internos ou externos autorizados.
O projecto descrito neste documento, denominado por Projecto em Engenharia Informática (PEI), está integrado no Mestrado em Engenharia Informática (MEI) ministrado pela Faculdade de Ciências da Universidade de Lisboa (FCUL) e foi desenvolvido autonomamente com supervisão numa grande empresa directamente relacionada com sistemas e telecomunicações.
Pretende-se com este projecto melhorar a segurança da informação no que diz

respeito à ocorrência de *malware* propagado automaticamente, através de uma rigorosa investigação e consequente desenvolvimento de um sistema de *software*. A integração num projecto de maior dimensão que está em produção na mesma empresa é uma futura meta do sistema de *software* desenvolvido e descrito neste documento.

1.1 Motivação

"A divisão entre o consumidor e a empresa está a desaparecer pouco a pouco. Os consumidores também são funcionários, e os funcionários conduzem negócios pessoais nas próprias estações de trabalho ou até mesmo nos seus laptops pessoais ligados à rede interna da empresa"

A rápida detecção da propagação automática de *malware* (*worms* e *bots*), em grandes redes empresariais e a sua correcta identificação e monitorização é essencial para garantir a sua rápida contenção, remoção, e deste modo contribuir significativamente para a segurança da informação em grandes empresas.

Quando não identificado e contido atempadamente um *worm* ou um *bot* pode ter um impacto devastador na operacionalidade dos sistemas de uma rede empresarial e, consequentemente, no negócio dessa empresa, criando quebras significativas de produtividade e, em casos extremos, afectando negativamente o serviço prestado aos seus clientes.

Nas grandes empresas são utilizados mecanismos de segurança tradicionais (*firewalls, IDS*, etc) que são eficientes para ataques com origem externa não autorizada ou dito por outras palavras, ataques que partam de utilizadores externos com credenciais inválidas para acesso aos serviços da empresa, mas são ineficientes noutras situações que efectivamente podem ser bastante perigosas. Tipicamente estes mecanismos de segurança tradicionais são bastante ineficientes para ataques despoletados a partir das máquinas dos utilizadores (internos ou externos) com credenciais válidas para acesso aos serviços.
Estes utilizadores, quando internos, podem fazer parte de uma equipa de desenvolvimento, operações ou administração e muitas vezes até estagiários ou colaboradores temporários em regime de *outsource*. No caso dos utilizadores externos autorizados, uma empresa tem tipicamente os seus clientes, vendedores e operadores com credenciais válidas para acesso.

É fácil constatar que existe aqui uma elevada complexidade de gestão ao nível da protecção/segurança da informação interna da empresa, tendo em conta a heterogeneidade dos perfis dos seus utilizadores. Temos o exemplo do estagiário (na equipa de administração de sistemas) que leva para a empresa o seu *laptop*

sem qualquer anti-*malware* instalado, fazendo parte de uma *botnet*. Lógicamente a ligação deste *laptop* à rede interna da empresa certamente que a comprometiria, poderia por exemplo provocar uma interrupção nos serviços (através de um ataque de negação de serviços) ou furtar informação sensível que traria certamente custos elevados à empresa e afectaria negativamente todo o seu processo de negócio.

Existe assim a necessidade de criar um modelo/solução pouco intrusivo (ou completamente passivo), eficiente e de baixo custo mais adequado a este tipo de situações que, caso ocorram, os danos provocados (que numa situação normal colocariam em risco os serviços prestados pela empresa) poderiam ser atempadamente detectados e mitigados.

1.2 Objectivos

Os objectivos deste projecto deverão consistir na investigação, desenho, desenvolvimento e experimentação de uma solução que permita monitorizar e conter atempadamente *malware* propagado automaticamente (*worms* e *bots*) pela rede interna de grandes organizações.

A solução traduziu-se no desenvolvimento de um sistema de *software*, o *WMS* (capítulos 5 e 6), que envolveu a implementação de vários componentes com funções específicas e bem delineadas, como poderá ser visto em detalhe ao longo deste documento.

Este sistema permite capturar *malware* propagado automaticamente, recorrendo à integração de um *honeypot* nas sondas de rede *Pulso* que estarão estrategicamente distribuídas pela rede empresarial. Para além da tarefa de captura, os binários suspeitos de actividade maliciosa deverão ser identificados e analisados dinamicamente (alterações no sistema de ficheiros, entradas no registo, tráfego de rede, etc) recorrendo ao auxílio de plataformas externas altamente fiáveis e eficientes. O sistema deverá permitir ainda monitorizar um conjunto de métricas que posteriormente permitam avaliar, com base na informação recolhida pelas sondas de rede, diversos parâmetros que estão intimamente ligados a eventos de ataque que recorrem à propagação automática de *malware*.

Por fim o desenvolvimento deste projecto deverá ter em conta a possibilidade de integração na rede e portal do *Pulso* (capítulo 2) para efeitos de alarmística deste tipo de eventos de ataque.

1.2.1 Metodologia de Investigação

A metodologia que guiou toda a investigação inerente a este projecto foi a *Action Research*[1].
Trata-se de uma metodologia baseada num ciclo contínuo entre teoria e prática constituído por cinco fases (ver figura 1.1). De cada iteração do ciclo devem resultar aprendizagens e hipóteses de melhoria resultantes de tais aprendizagens. Ao longo deste relatório todas as fases deste ciclo serão implicitamente ou explicitamente descritas em pormenor, algumas delas, como já foi referido anteriormente foram iteradas, contribuindo assim para uma aprendizagem e consequente melhoria da solução.

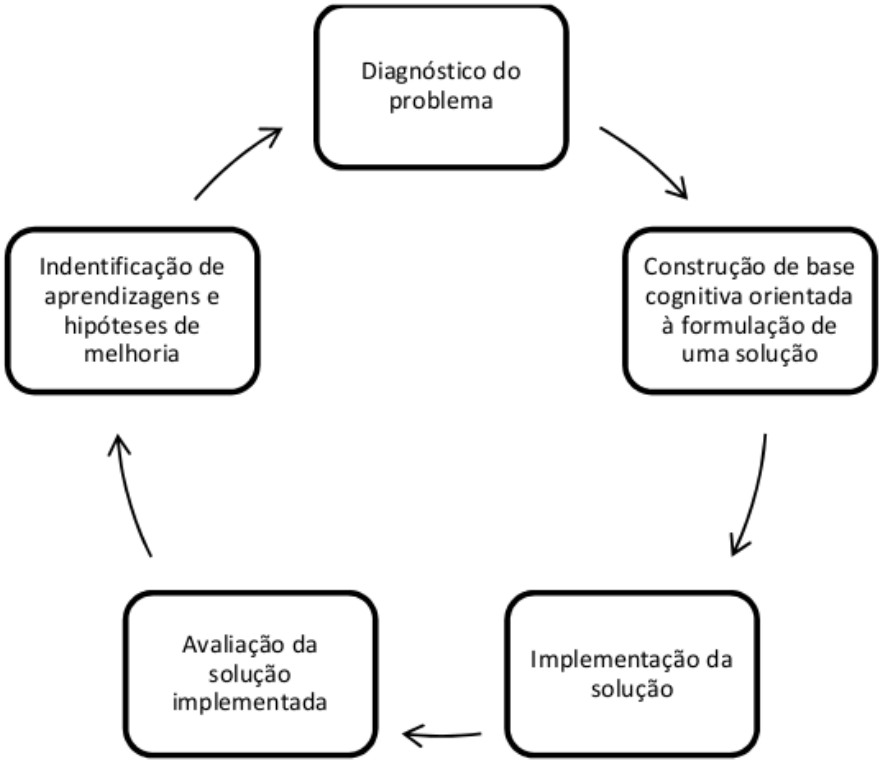


Figura 1.1: *Action Research*

1.2.2 Tarefas

- Investigar a teoria e prática (*i.e.* tecnologias, plataformas e projectos similares) relativamente às temáticas abordadas neste projecto (*i.e.* captura, identificação e análise automática de *malware* que envolva propagação automática, ou especificamente, os *worms* e *bots*).

[1] `http://www.web.net/robrien/papers/arfinal.html`

- Desenhar e implementar o *WMS*, um sistema distribuído composto por uma federação de sondas de rede *Pulso*. Estas sondas deverão comunicar com um servidor central que usufruirá de ligação à *Internet*.
  - Utilizar a federação de sondas de rede *Pulso* com o *dionaea honeypot*[2] para capturar tráfego suspeito de actividade maliciosa.
  - Utilizar, a partir do servidor central, as plataformas externas *VirusTotal*[3] e *Anubis*[4] que permitem através de serviços remotos identificar e analisar dinamicamente *malware*.
    * Para automatização da comunicação com o *VirusTotal* implementar um sub-componente de *software* (*avsubmit*) para tratar do processo.
    * Para automatização da comunicação com o *Anubis* implementar um sub-componente de *software* (*ansubmit*) para tratar do processo.
    * Implementar o sub-componente de *software* (*alarmwrite*) que tratará do processo de produção e escrita de alarmes de ocorrências de ataque via *malware* propagado automaticamente.
- Implementar os sub-componentes de *software* (*receivehttp*, *mysql* e *incident-model*) para recepção das *fingerprints*, definição do estado do evento de ataque a partir de cada sonda de rede *Pulso* e registo automático das *fingerprints* assim como a salvaguarda do binário suspeito envolvido em cada ataque.
- Implementar todo o sub-sistema de *back-end* (*mwmonitor*) no servidor central para controlar a selecção de binários maliciosos e os sub-componentes de *software* responsáveis pela identificação com o *VirusTotal* (*avsubmit*), análise com o *Anubis* (*ansubmit*) e alarmística (*alarm* e *alarmwrite*).
- Idealizar e construir métricas de segurança que permitam monitorizar a *Quality of Protection (QoP)* da rede no que diz respeito às ocorrências de eventos de ataque através de *malware* propagado automaticamente.
- Implementar uma aplicação *web* (*WMSi*), para monitorização e/ou visualização gráfica das estatísticas e tendências dos eventos de ataque e/ou das ocorrências de *malware* propagado automaticamente.
- Implementar aplicações extra (*mailrobot* e *artifactsrobot*) que utilizem os serviços do sub-sistema de *back-end* do servidor central (sub-sistema *mwmonitor*).
- Configurar o *WMS* para efeitos de experimentação e avaliação da solução.

[2] `http://dionaea.carnivore.it/`
[3] `http://virustotal.com/`
[4] `http://anubis.iseclab.org/`

- Avaliar experimentalmente o *WMS* (*i.e. footprinting*, estatísticas, tendências, etc) com tráfego intenso (*i.e.* na *Internet*).

Como se pode verificar, estas tarefas instanciam implicitamente ou explicitamente todas as fases da metodologia de investigação utilizada (ver figura 1.1), e foram totalmente planeadas no início de desenvolvimento do projecto.

1.3 Empresa e equipa de trabalho

A Portugal Telecom (PT)[5] é uma multinacional fundada em 1994, actualmente o maior provedor de telecomunicações em Portugal.

A PT Comunicações (PTC)[6] é uma empresa do Grupo PT, criada em 18 de Setembro de 2000, que dispõe desta infra-estrutura para proporcionar aos seus clientes mais e melhores meios de comunicação.

Para aumentar os índices de produtividade e competitividade tendo em conta a constante evolução das tecnologias de informação e a preocupação com aspectos que garantam a sua eficiência, disponibilidade e segurança foi criado a o departamento de Eficiência, Disponibilidade e Segurança dos Sistemas de Informação e/ou Tecnologias de Informação (EDS-SI/TI), onde actualmente, se está a desenvolver o sistema *Pulso*.

1.4 Estrutura do documento

No primeiro capítulo é discutida a problemática da segurança de informação em que se enquadra este projecto.

Depois é feito um enquadramento da organização em que o projecto se insere, para efeitos de futura integração num sistema de grande dimensão que se encontra em produção, o *Pulso* (capítulo 2). De seguida é apresentada toda a teoria inerente às tecnologias e mecanismos usados no projecto desenvolvido, nomeadamente no que diz respeito à captura, identificação e análise de *malware* propagado automaticamente (capítulo 3).

No capítulo 4, são referidas e detalhadas as plataformas de *software* de terceiros utilizadas neste projecto. Neste documento é ainda detalhado todo o modelo teórico do projecto. Esse modelo teórico deu origem a um sistema de *software*, o *WMS* (capítulo 5). No capítulo 6, o *WMS* é avaliado experimentalmente através da instanciação de alguns cenários de utilização (monitorização através da aplicação web *WMSi*) e é realizada uma discussão geral do projecto incluindo

[5] `http://telecom.pt/`
[6] `http://ptcom.pt/`

por exemplo, a discussão de algumas abordagens tecnológicas alternativas que foram investigadas (capítulo 7).

Por fim conclui-se este documento apresentando algumas ideias para trabalho futuro (capítulo 8).

Capítulo 2

Sistema *Pulso*

A equipa responsável pela gestão do risco técnico, segurança e qualidade técnica dos sistemas e tecnologias de informação da PTC desenvolveu um programa de projectos denominado por *Pulso* [1]. O termo *Pulso* refere-se simultaneamente ao programa de projectos, à plataforma técnica que o alberga e ao portal que o suporta e lhe dá a face externa agregando análises de *Quality of Maintenance (QoM)*, QoP e *Quality of Service (QoS)*. Para tal serve-se de informação obtida a partir de várias fontes (computadores que disponibilizem serviços importantes, bases de dados, infra-estrutura de rede, *call-center's* e lojas) para análise estatística e monitorização de diversos parâmetros. O sistema adicionalmente gera e envia alarmes às entidades que aparentam estar num estado ou situação fora do normal. A plataforma *Pulso* suporta a prevenção de incidentes informáticos assim como a rápida detecção e reacção a estes, na eventualidade dos mecanismos de prevenção terem sido insuficientes.

Em suma pode-se afirmar que o *Pulso* é uma plataforma desenvolvida para a detecção, colecta, análise e comunicação de eventos técnicos relevantes ao nível da qualidade, risco técnico e segurança dos principais sistemas e infra-estruturas informáticas (SI/TI) de suporte ao funcionamento regular da empresa.

2.1 Objectivos

O *Pulso* pretende obter de forma contínua, cíclica e automática, o *scoring* de risco técnico de todos os sistemas de informação mais importantes e respectivas infra-estruturas tecnológicas de suporte, relativamente à sua *QoS*, *QoP* e *QoM*. Outro dos objectivos do *Pulso* consiste na detecção e reacção em tempo útil de ocorrência de incidentes relevantes ao nível de falhas técnicas, ataques externos ou internos (onde o projecto descrito neste documento se enquadra) e uso abusivo de recursos. Ao nível da interface com a organização e ao nível da interface com os principais responsáveis operacionais, pretende-se através de um Portal Colaborativo

e Operacional (a face visível do sistema *Pulso*), ter um meio formal, organizado e centralizado, de disponibilização de conteúdo para a função de gestão de risco técnico, segurança e controlo dos sistemas de informação.

2.2 Arquitectura

A arquitectura do sistema *Pulso* tem por base um subsistema distribuído para medir os parâmetros de qualidade da rede entre locais considerados críticos para a PTC. Tendo em conta a assimetria e a heterogeneidade da distribuição geográfica dos locais de acesso e dos sistemas numa empresa como a PT foram definidos três tipos de componentes: as sondas de rede (*collectors*), os agentes de sistema (*targets*) e os *loggers* (ver figura 2.1).

Os agentes são colocados em um ou mais sistemas cujo *QoS* se pretende monitorizar, ou seja em locais que sejam considerados relevantes para o negócio (*i.e.* tendo em conta o número de utilizadores e aplicações utilizadas).

Cada sonda de rede efectua medições relativamente aos agentes de sistema monitorizando e medindo parâmetros de rede na ligação lógica compreendida entre ela e cada agente, note-se que cada agente apenas responde aos pedidos de medição das sondas.

Os *loggers* têm a missão de recolher periodicamente os resultados das medições que estão armazenados em máquinas chamadas *datasources* (máquina que armazena temporariamente informações do sistema *Pulso*). No caso da medição de rede os *datasources* correspondem às sondas de rede. O armazenamento dos resultados das medições para a base de dados do *Pulso* é também uma tarefa dos *loggers*.

Cada sonda efectua a(s) medição(ões) para o(s) agente(s) para o(s) qual(is) foi programado e armazena os eventos localmente no formato canónico do *Pulso*. Cada sonda está registada como *datasource* no *Pulso* e por isso o módulo de recolha armazena os eventos medidos no repositório de eventos do *Pulso*. Existe ainda um componente que permite co-relacionar a informação recolhida com acontecimentos de um número mais elevado de abstracção (*Real Time Correlation Engine*) para depois ser processada em relatórios, gráficos e modelos estatísticos (*Analytical Forecasting Engine*) que compõem o portal *Pulso* com vários indicadores de medida para efeitos de monitorização. Por fim são ainda produzidos e enviados para a equipa técnica, alarmes a reportar todas as situações críticas ocorridas.

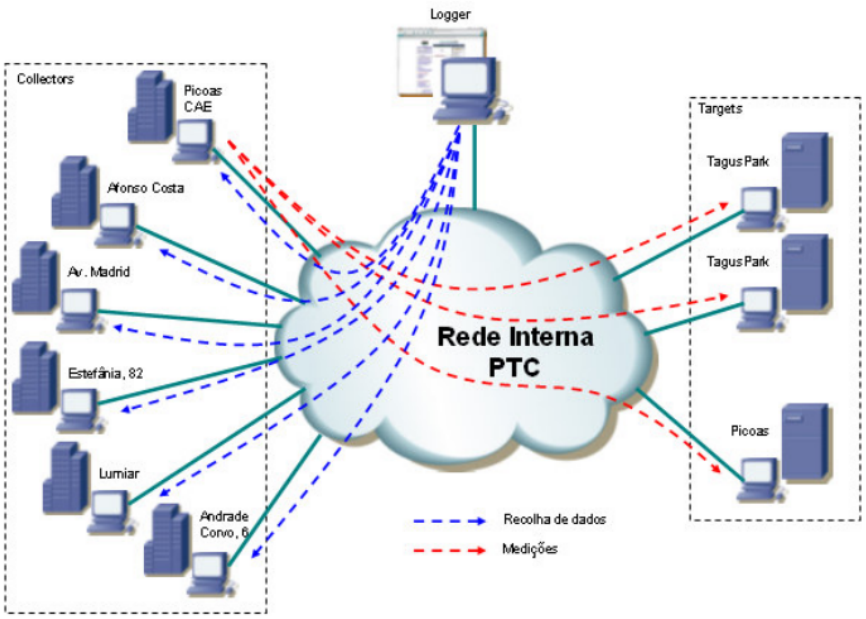


Figura 2.1: Arquitectura de *collectors*, *targets* e *loggers*

2.3 Resumo

O sistema *Pulso* da PT constitui uma solução 100% *open source* que tem demonstrado ser bastante eficaz e de muito baixo custo, capaz de enfrentar os elevados desafios propostos pelas organizações que nem sempre estão dispostas a alterar os seus métodos tecnológicos internos em prol da segurança de informação. O processo de monitorização passiva (não intrusiva) do (in)correcto funcionamento de sistemas, protocolos de rede e bases de dados que o sistema *Pulso* dispõe oferece uma solução completa para gestão de risco técnico, segurança e controlo de sistemas de informação.

Muito coisa haveria ainda para falar do sistema *Pulso*, no entanto o objectivo não passa por detalha-lo em pormenor, caso assim o fosse certamente que se iria sair do perímetro deste projecto, neste caso optou-se por resumir os objectivos e a arquitectura. Desta forma fica-se com uma noção do que se trata o *Pulso*, sistema este que poderá no futuro servir-se do sistema construído e apresentado em detalhe neste projecto.

Capítulo 3

Captura, Identificação e Análise de *malware*

Segundo o projecto *Honeynet*[1] e os *Kaspersky Labs*[2] a combinação de técnicas que permitam a captura de *malware* propagado automaticamente, a identificação do binário malicioso e a respectiva execução virtualizada ou análise dinâmica, são tarefas fundamentais que permitem ter conhecimento da denominação do *malware* (identificação) assim como de todo o seu processo de ataque e infecção (análise).

Neste capítulo é descrita cada uma destas actividades nas suas vertentes mais teóricas, incluindo por vezes algumas instanciações com exemplos práticos. Além de ser dada uma panorâmica geral acerca de algumas categorias de *malware* mais relevantes, o foco incide nas categorias de *worms* e *bots*, que na prática são as categorias de *malware* suportadas por este projecto tendo em conta a tarefa de captura recorrendo à utilização de *honeypots*.

3.1 *Malware*

Malware [13][18][38] é um termo colectivo dado a *software* malicioso que se apodera de um sistema sem autorização do seu utilizador. No mundo da computação uma das grandes preocupações hoje em dia está relacionada com este tipo de *software*.

O volume e a complexidade das técnicas de ataque realizadas por *malware* são actualmente bastante críticas e têm tendências para aumentar nos próximos anos, nomeadamente com a evolução da computação móvel e na nuvem (*cloud computing*).

[1] `http://honeynet.org/`
[2] `http://kaspersky.com/`

3.1.1 Objectivo

O *malware* foi criado especificamente com o objectivo de danificar sistemas computacionais sem o conhecimento do respectivo utilizador. Inclui vírus, *trojan horses*, *spyware*, *worms* e *bots* que serão descritos nas próximas secções, pois cada uma destas categorias de *malware* inclui características e propósitos específicos que as diferem umas das outras. Depois ainda existe *software* que permite auxiliar o *malware* na evasão de anti-*malware*, *firewalls* e *IDS*. Alguns especialistas consideram que este tipo de *software* é considerado *malware*, no entanto ele é aqui descrito como *software* que permite camuflar *malware*, dentro desta classe encontram-se principalmente os *rootkits* e os *packers*.

Este projecto cobre apenas as categorias de *malware* que envolvem propagação automática numa rede de computadores, ou seja os *worms* e os *bots*, daí a ser dada mais atenção a estas duas categorias.

3.1.2 Vírus

Trata-se de um programa que se anexa a outro programa para o infectar e executar funções indesejadas. Alguns são de fácil detecção e remoção, outros são mais avançados. Temos para este último caso o exemplo dos vírus que usam mecanismos polimórficos (alterações), prolongando desta forma a sua permanência no sistema da vítima e evitando ao máximo a respectiva detecção por parte de um anti-*malware*. Um vírus tipicamente requer que o utilizador execute uma aplicação ou *script* infectado para que exista propagação pelo sistema de ficheiros. Tipicamente um utilizador nunca se apercebe que executou um vírus mas sim outra aplicação ou por exemplo uma imagem.

3.1.3 *Trojan Horse*

Um *trojan horse* simula o comportamento de um programa autêntico como por exemplo uma *login shell* para roubar a palavra chave de *login* com o objectivo de ganhar controlo remoto do sistema da vítima. Outras actividades como a monitorização de sistemas, corrupção dos recursos de sistema (como ficheiros ou dados do disco) e desactivação de serviços específicos, fazem parte do reportório de ataques típico desta categoria de *malware*.

3.1.4 *Spyware*

Spyware é o termo colectivo para *software* que monitoriza e recolhe informação pessoal acerca do utilizador (a vítima). Monitoriza e recolhe dados acerca das páginas *web* frequentemente visitadas, endereços de *e-mail*, números de cartões

de crédito, teclas pressionadas pelo utilizador, etc. Geralmente esta categoria de *malware* entra no sistema quando é feito *download* de *trial* ou *free software*.

3.1.5 *Worm*

Um *worm* [13][37][44] é um programa capaz de se propagar automaticamente através de redes de computadores, enviando cópias de si mesmo de computador para computador. Propaga-se através da exploração de vulnerabilidades de segurança existentes em *software* e ao contrário de um vírus não se anexa a ficheiros nem necessita de execução explícita para se propagar. Um *worm* tipicamente não infecta ficheiros nem corrompe informação, no entanto pode consumir vários recursos de rede degradando-a gradualmente ao longo do tempo, assim como lotar um disco rígido quando envia várias cópias de si mesmo para outras máquinas vulneráveis na rede. Temos como exemplos mais sonantes o *Sasser, MyDoom, Blaster, CodeRed, Melissa, Conficker* e o *Stuxnet*.

Características e estratégia

Num ataque automatizado [10] o pirata informático desenvolve um ou mais programas que exploram uma ou mais vulnerabilidades. Os programas não precisam ser desenvolvidos de raiz, podendo ser utilizados componentes previamente existentes ou até *toolkits* (*Metasploit Framework*[3], *WormGen*, etc) próprios para esse fim.

Os *worms* atacam vulnerabilidades de *software* e a operação mais comum hoje em dia, consiste em deixar um *bot/zombie* no computador da vítima. Muitos programas maliciosos não têm nenhum objectivo especificamente criminoso mas são criados para visarem apenas a fama dos seus criadores. No entanto, muitas vezes acabam por criar instabilidades nos computadores em que penetram ou na rede onde circulam. Foi o caso do famoso *Morris worm* que não fazia nada de mal intencionalmente, mas cuja propagação parou a *Internet*. Convém notar que um ataque automatizado pode ser dirigido. Recentemente a tendência no mundo dos *worms* parece ser a dos que infectam um número limitado de computadores todos os dias (*i.e.* alguns milhares), de modo a evitar tornarem-se conspícuos. Um *worm* como se propaga automaticamente de computador para computador, embebe um selector de alvos e um motor de varrimento que realizam automaticamente algo de equivalente às fases de *footprinting*, varrimento de portas e enumeração de serviços vulneráveis. O selector de alvos pode fazer algo tão simples como sortear um conjunto de endereços *IP (Internet Protocol)* ou obter os endereços de correio electrónico do livro de endereços da vítima onde está a ser executado no mo-

[3] `http://metasploit.com/`

mento.
O motor de varrimento verifica se os alvos seleccionados têm a(s) vulnerabilidade(s) que o *worm* utiliza para se propagar.
No *worm*, a fase de identificação de vulnerabilidades é limitada ou mesmo inexistente, pois este tem um conjunto limitado de vulnerabilidades que pode explorar.
A ogiva do *worm* é o *exploit* que realiza o ataque. Normalmente um *worm* não faz elevação de privilégios, nem é necessário, mas instala um *rootkit* ou causa algum tipo de estrago na vítima: a carga transportada pelo *worm*.
Um *worm* tem ainda um motor de propagação na rede, responsável por controlar a sua transferência de computador para computador.
Na figura 3.1 pode-se visualizar o esquema de ataque automatizado para o caso de um *worm*.

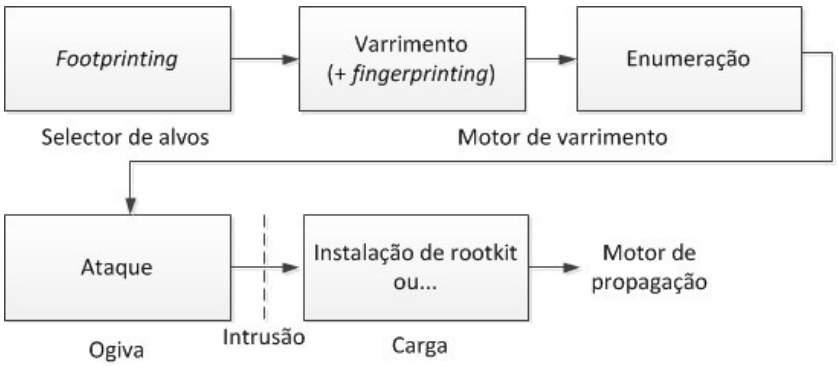


Figura 3.1: Fases de um ataque automatizado

Estrutura genérica

- **Selecção de alvos e varrimento**
  Esta á primeira fase de propagação do *worm* cujo objectivo principal consiste em descobrir ou detectar novas máquinas para infectar. Uma máquina vulnerável pode ser descoberta pelo *worm* de várias formas, por pesquisa, por lista de alvos ou por monitorização passiva. Os *worms* mais destrutivos recorrem tipicamente a combinações destas técnicas, podendo desta forma tirar o melhor partido de cada uma.

  1. **Pesquisa**
     Esta técnica consiste no varrimento de um conjunto de endereços *IP* até detectar algum pertencente a uma máquina vulnerável. As formas mais simples de pesquisar máquinas vulneráveis baseia-se na pesquisa sequencial e pesquisa aleatória.
     A pesquisa sequencial consiste em pesquisar máquinas vulneráveis a

partir de um bloco de endereços *IP* conhecidos à priori.
Por outro lado a pesquisa aleatória consiste em pesquisar máquinas vulneráveis a partir de uma semente pré-definida com o propósito de geração de endereços *IP* aleatórios.

2. **Lista de alvos**
   A descoberta de máquinas pode ser conseguida recorrendo a uma lista de alvos. Os *worms* que utilizam estas listas para atacar máquinas vulneráveis são também chamados de *hitlist worms* e são caracterizados pela sua rápida propagação.

   – **Lista de alvos pré-gerada**
     Nesta lista tipicamente embebida no *worm*, consta um conjunto de endereços *IP* de máquinas conhecidas ou suspeitas de determinada vulnerabilidade de segurança. Uma lista pequena deste tipo pode ser usada para acelerar o processo de propagação.

   – **Lista de alvos gerada externamente**
     Esta lista não está embebida no *worm* pois é mantida por um servidor dedicado. O *download* da lista de endereços *IP* é realizado pelo *worm* para infectar as máquinas correspondentes.
     O facto destas listas estarem armazenadas em servidores dedicados faz com que a actualização dos endereços *IP* de máquinas vulneráveis seja mais simples e frequente, mas em contrapartida se o servidor for comprometido pode-se bloquear facilmente todo o processo de propagação de um *worm* que utilize este método.

   – **Lista de alvos interna**
     Neste caso o *worm* utiliza a informação armazenada (endereços de *e-mail*, etc) na máquina infectada de modo a decidir que máquinas irá atacar de seguida. Os *worms* que utilizam uma lista de alvos interna ou baseada na máquina alvo denominam-se por *topological worms*.

3. **Monitorização passiva**
   Os *worms* que recorrem à monitorização passiva não procuram activamente por novas máquinas vulneráveis mas ficam à espera que estas surjam na rede ou então confiam que a vítima vá descobrindo novas

máquinas vulneráveis através de possíveis ligações inocentes à *Internet*.

- **Ogiva (activação do *worm*)**
  - **Activação humana**
    Os *worms* que necessitam de activação humana normalmente não se propagam muito rápido, na maioria das vezes o *worm* é activado (ogiva) depois da vítima fazer *login/logout* no sistema infectado.
  - **Activação baseada em escalonamento**
    Este processo de activação de *worms* torna a propagação bem mais rápida em relação ao método de activação referido anteriormente.
    Temos por exemplo o caso dos *updates* automáticos que podem ser usados para instalar e executar *malware*, neste caso *worms*.
  - **Auto-activação**
    Grande parte dos *worms* que se propagam mais rápido utilizam esta técnica de auto-activação.
    Este método é conseguido através da exploração de vulnerabilidades de um serviço em execução.
- **Carga**
  A carga ou *payload* do *worm*, é o código que o *worm* transporta que não está relacionado com o código que permite a propagação deste numa rede. Esta carga pode variar dependendo do autor e do objectivo do *worm*.
- **Propagação e mecanismos de distribuição**
  Serão apresentados de seguida, muito resumidamente, três métodos típicos levados a cabo por um *worm* para se transferir na rede de um computador para outro.
  - **Auto-Transporte**
    Um *worm*, no que diz respeito ao mecanismo de propagação, diz-se de auto-transporte se este se transmite a si mesmo como processo de infecção. Este mecanismo é tipicamente usado quando o ataque inicial é imediatamente seguido pela transmissão do *worm* e respectiva activacao na máquina comprometida.
  - **Segundo canal**
    Alguns *worms* requerem um segundo canal de comunicação para completar o processo de infecção. O exemplo clássico é a máquina da

vítima pedir transferência remota do *worm* para completar o processo de infecção.

Conficker

O *Conficker*[4] (ver figura 3.2), foi detectado pela primeira vez em 2008 e também é conhecido por *Downup*, *Downadup* e *Kido*. Trata-se de um *worm* que tem como objectivo afectar máquinas que tenham instalado o sistema operativo *Microsoft Windows*. O *worm* propaga-se e explora uma vulnerabilidade (*MS08-067*) presente no serviço de *RPC* do *Microsoft Windows*. Acredita-se que o *Conficker* tenha sido o *worm* que mais se propagou pela *Internet* desde o *SQL Slammer*, em 2003. A razão da rápida propagação inicial do *worm* tem a ver com o elevado número de computadores que utilizam o sistema operativo *Microsoft Windows* (estimado em 92.12% de utilizadores em todo o mundo, que ainda necessitam aplicar as actualizações da *Microsoft* para a vulnerabilidade *MS08-067*. Em 2009, o número estimado de computadores infectados variou entre 9 e 15 milhões, o que mostra bem o poder de propagação deste *worm*.

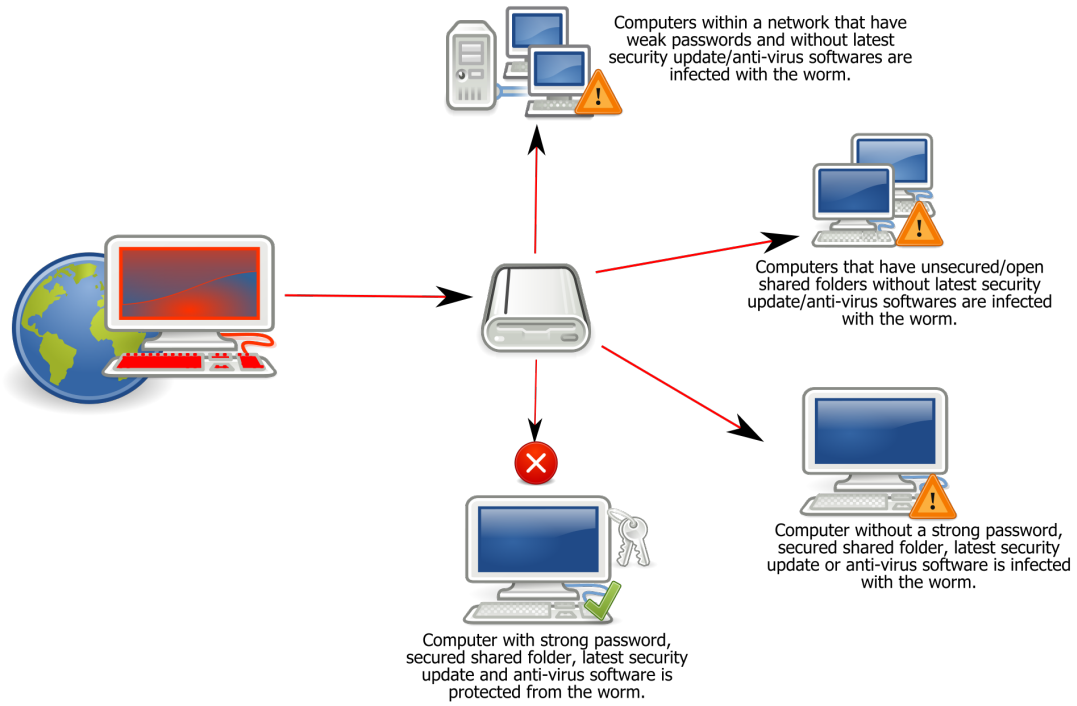


Figura 3.2: *Conficker*

[4] `http://mtc.sri.com/Conficker/`

Stuxnet

O *Stuxnet* [11] é um *worm* desenvolvido especificamente para atacar o sistema operativo *SCADA* que foi desenvolvido pela *Siemens* para controlar as centrífugas de enriquecimento de urânio iranianas. Foi descoberto em Junho de 2010 por uma empresa de segurança bielorussa assim como foi o primeiro *worm* descoberto, que espiona e reprograma sistemas industriais.

3.1.6 *Bot/Botnet*

De modo similar ao *worm*, um *bot* [10][13][16] é um programa capaz de se propagar automaticamente explorando vulnerabilidades existentes ou determinadas falhas na configuração de *software* instalado no computador.
Adicionalmente um *bot*, dispõe de mecanismos de comunicação com o atacante, permitindo assim que seja controlado remotamente. Este controlo é realizado tipicamente através do *Internet Relay Chat (IRC)*, ou seja, o *bot* liga-se a um servidor de *IRC* e entra num determinado canal, esperando pela ligação e instruções do atacante no mesmo servidor/canal. O atacante envia para o canal mensagens compostas por sequências especiais de caracteres que são interpretadas pelo *bot*. Estas mensagens mandam desferir ataques na *Internet* com propósitos de negação de serviços, agir como *spyware*, enviar *spam* e *e-mails* de *phishing*.
Uma *botnet* é uma rede formada por computadores infectados por *bots*. Estas redes podem ser compostas por centenas ou milhares de computadores, e um atacante que tenha controlo sobre uma *botnet* pode utiliza-la para aumentar a potência dos seus ataques, por exemplo, para enviar centenas de milhares de *e-mails* de *phishing* ou *spam*, desferir ataques de negação de serviço, etc.
Temos como exemplos mais sonantes, o *Storm worm*[5] ou o *Kolab* que apesar de serem normalmente chamados de *worms*, são na realidade sistemas distribuídos bastante complexos (*botnets*) que têm evoluído ao longo do tempo, e que são usados pelos seus criadores para fornecer um conjunto de serviços de criminalidade informática como furtar informação sensível, números de cartões de crédito, etc.

Kolab

O *Net-Worm:W32/Kolab*[6] é um *worm* que funciona como um *IRC bot*, ou seja entra na categoria dos *bots*. Permite ao atacante usar o computador da vítima como um *proxy* ou como uma ferramenta para executar ataques de negação de serviço. Adicionalmente este *bot* age ainda como *spyware*, furtando informação relevante

[5] `http://www.cyber-ta.org/pubs/StormWorm/report/`
[6] `http://securelist.com/en/descriptions/24390763/Net-Worm.Win32.Kolab.acem`

do computador da vítima. Permite realizar download de *malware* adicional e dar ao atacante controlo parcial do computador da vítima.

3.1.7 Técnicas de evasão de *software* anti-*malware*

Rootkit

Um rootkit [7] é um *software* dissimulado que esconde outro *software* dissimulado. O seu objectivo principal consiste em realizar operações nefastas como por exemplo deixar um *bot* ou uma *backdoor* para futuro acesso ao sistema. Um bom *rootkit* providencia sempre acesso posterior ao pirata informático mesmo que a máquina da vítima seja reiniciada. Normalmente um *rootkit* esconde as pistas da sua presença automaticamente, isto é, ficheiros, entradas de registo, informações de tráfego de rede e mais importante ainda, esconde-se a si mesmo no sistema, enganando assim um anti-*malware* mais facilmente.

Packer

Um *packer* normalmente é usado para comprimir ficheiros *Windows PE* (*Portable Executable*) maliciosos de maneira a confundir o *software* de anti-*malware*.
Os *packers* mais comuns são o *UPX*[7], o *ASPack*[8] e o *tElock*[9] e estes podem ser usados várias vezes e intercalados para melhorar o processo de compressão alterando mais eficazmente a estrutura interna dos ficheiros.

3.2 Captura de *malware*

3.2.1 *Honeypots*

Um *honeypot* [10][16][29][32][33][39] é uma armadilha para piratas informáticos e/ou *malware* que se propaga automaticamente (*worms* e *bots*). É composto por um ou mais computadores que dão ideia de pertencerem a um sistema de produção, no entanto na realidade, estão isolados e parcialmente desprotegidos, embora contenham mecanismos de monitorização que regista informação sobre intrusões que possam vir a ocorrer. Tipicamente utiliza-se o termo *honeynet* a um *honeypot* composto por vários computadores.
Como já foi referido um *honeypot* não faz parte de um sistema em produção, consequentemente qualquer tráfego para ele dirigido pode ser considerado suspeito de actividade maliciosa.

[7] `http://upx.sourceforge.net/`
[8] `http://aspack.com/aspack.html`
[9] `http://telock.softpedia.com/`

Alta interactividade

Os *honeypots* de alta interactividade oferecem um sistema operativo completo, bem como aplicações reais, com o qual o atacante pode interagir livremente. Estes não emulam serviços, são computadores reais com aplicações reais prontas a serem comprometidas. As vantagens são evidentes, não só são capazes de detectar atacantes que varrem o sistema à procura de vulnerabilidades mas também permitem que os atacantes invadam o sistema e tomem controlo total do mesmo. Sendo assim, é possível capturar, entre outros, *rootkits* (dos quais os atacantes normalmente fazem *upload* para o sistema comprometido), analisar o que os atacantes digitam e monitorizar todas as suas comunicações com outros sistemas. Como resultado, é possível aprender quais os motivos que levam um atacante a comprometer o sistema, o seu nível técnico e demais informação. Visto que *honeypots* de alta interactividade não emulam serviços, então permitem que se capturem ataques desconhecidos ou comportamentos inesperados. Contudo, todas estas vantagens têm o seu preço. *Honeypots* de alta interactividade implicam um risco elevado, uma vez que os atacantes têm acesso a um sistema operativo real que pode ser usado para atacar outros sistemas. Além disso, estes são muito mais complexos que os *honeypots* de baixa interactividade.
É necessário que se construa e configure um sistema real com os quais os atacantes possam interagir, onde a complexidade será cada vez maior à medida que tentamos minimizar os riscos inerentes à possibilidade do nosso *honeypot* ser usado para atacar outros sistemas.
Como exemplo de um *honeypot* de alta interactividade temos o *Symantec's Decoy Server*[10].

Média interactividade

Este tipo de *honeypots* não pretende simular um sistema operativo completo nem implementar todos os detalhes dos protocolos de aplicação. O que eles fazem é fornecer respostas suficientes que *exploits* conhecidos esperam em determinadas portas ao tentar interagir com determinados serviços. Contudo, estes *honeypots* têm de fornecer algum tipo de sistema de ficheiros virtual, bem como alguns utilitários *standard* do sistema operativo que se pretende emular. Os *honeypots* de média interactividade têm demonstrado ser os mais eficientes a colecionar autonomamente *malware*. A grande maioria do *malware* espalha-se usando padrões similares, *exploits* comuns, *shellcodes* comuns e *shell scripts* comuns. Apesar destes *honeypots* poderem ser facilmente detectados através de diversas técnicas, isto acaba por ser um factor pouco importante, visto que a maioria do *malware* espalha-

[10] `http://symantec.com/`

se de forma automatizada, não havendo um prévio reconhecimento do sistema a atacar.
Como exemplos mais sonantes temos o *mwcollectd*[11] e principalmente o *dionaea* que foi o escolhido para este projecto e que será detalhado mais à frente no capítulo 4.

Baixa interactividade

Os *honeypots* de baixa interactividade normalmente emulam sistemas e serviços para que os atacantes possam interagir com eles. Contudo, os atacantes estão bastante limitados naquilo que podem fazer. Normalmente, podem-se conectar e executar apenas alguns comandos básicos. Estes *honeypots* de baixa interactividade são também mais simples de instalar e configurar e ao mesmo tempo existe um risco menor associado, uma vez que os serviços emulados limitam o que um atacante pode ou não fazer. Quando um atacante executa algo de que o *honeypot* não está à espera, o último tem dificuldade em perceber as acções do atacante e não é capaz de responder adequadamente ou capturar a sua actividade.
Alguns exemplos de *honeypots* de baixa interactividade incluem o *nepenthes*[12][43] e o *honeyd*[13][33].

Baixa Interactividade	**Média Interactividade**	**Alta Interactividade**
Fácil de instalar e configurar	Requerem conhecimentos sólidos para instalação e configuração	A sua instalação e configuração são bastante complexas
Risco reduzido, os serviços emulados limitam o que o atacante pode fazer	Risco reduzido, os serviços emulados limitam o que o atacante pode fazer	Risco elevado, já que os atacantes têm um sistema operativo real com o qual podem interagir
Captura uma quantidade de informação muito limitada	Excelente para a captura de *malware*	Permite a captura de ferramentas, comunicações e *keystrokes*

Tabela 3.1: Tabela comparativa dos níveis de actividade de *honeypots*

Vantagens

- **Os *honeypots* reduzem os falsos positivos**
  Este tem sido o maior desafio da maioria das tecnologias de detecção, visto que as mesmas geram imensos alertas falsos (este problema é similar aos alarmes contra roubo).

- **Os *honeypots* coleccionam pequenas quantidades de dados**
  Visto que os *honeypots* apenas recolhem dados quando algo interage com

[11] `http://code.mwcollect.org/`
[12] `http://nepenthes.carnivore.it/`
[13] `http://honeyd.org/`

eles a quantidade de dados recolhidos acaba por não ser muita. Quando dispomos de centenas de *Mega Bytes* (*MB*) de alertas por dia torna-se óbvio que uma grande quantidade escapará a uma análise cuidada.
O facto dos *honeypots* coleccionarem poucos *MB* de dados torna-os muito mais fáceis de gerir.

- **Os *honeypots* podem detectar falsos negativos**
  Outra falha habitual das tecnologias tradicionais é a falha na detecção de ataques desconhecidos, visto que dependem de assinaturas conhecidas. Ou seja, essas tecnologias implicam que antes alguém tem de já ter sido comprometido.
  Os *honeypots* por outro lado são desenhados para detectar e capturar ataques desconhecidos, qualquer actividade detectada pode revelar novos ataques.

- **Os *honeypots* não requerem grandes recursos**
  Basicamente, qualquer máquina antiga é suficiente para a instalação e posterior configuração de um *honeypot*.

- **Os *honeypots* podem capturar dados cifrados**
  Mesmo que um ataque seja levado a cabo com recurso a cifragem o *honeypot* pode capturar esta actividade. Visto que o *honeypot* será um dos nós finais onde a decifragem é feita, isto acaba por não ser problema.

Limitações

- **Os *honeypots* implicam risco**
  Obviamente, cada vez que colocamos um *honeypot* em produção introduzimos um risco adicional na rede informática. Ou seja, existe sempre o perigo de um atacante tomar conta do sistema e usá-lo para lançar ataques contra outros sistemas, internos ou externos.

- **Os *honeypots* têm um campo de visão limitado**
  Ou seja, apenas vêm aquilo que interage com eles. Isto poderá, contudo, ser visto como uma vantagem, dependendo da perspectiva.

3.3 Identificação/detecção e análise de *malware*

As técnicas usadas para identificação/detecção [18][41] de *malware* podem ser classificadas em duas categorias:

- detecção baseada em assinaturas.

- detecção baseada em anomalias.

A técnica de detecção baseada em anomalias recorre a determinado conhecimento (chamadas ao *kernel*, etc) que permite deduzir se um programa se trata ou não de *malware*.
A detecção baseada em especificações é uma técnica especial de detecção baseada em anomalias. Este tipo de detecção baseada em especificações recorre a um conjunto de regras que define a malícia de um programa. Um programa que viole o conjunto de regras é considerado *malware*.

3.3.1 Identificação/detecção de *malware*

Técnicas de identificação/detecção de *malware*

- **Identificação/detecção baseada em assinaturas**
  Esta técnica tem a vantagem de ser bastante rápida mas está hoje bastante obsoleta pelo facto de apresentar os seguintes problemas:

  – Dificuldade na detecção de *malware* polimórfico (novas variantes a cada execução) e metamórfico (re-programação do comportamento através de técnicas de ofuscação).

  – As tarefas de extracção e distribuição de assinaturas são complexas.

  – O tamanho do repositório de assinaturas pode crescer a um ritmo alarmante.

  Para suprimir as lacunas existentes no método de detecção baseado em assinaturas surgiram os seguintes métodos de detecção:

- **Identificação/detecção baseada em especificações**
  Este método de detecção deriva do método de detecção baseado em anomalias. Neste caso existe uma fase de treino em que se tenta aprender todos os comportamentos válidos do programa ou sistema que precisa de ser inspeccionado. A maior limitação deste método prende-se com a dificuldade em especificar com exactidão o comportamento de um programa ou sistema.

- **Identificação/detecção baseada em comportamentos**
  Este método identifica as acções realizadas pelo *malware* em vez de se basear em *binary patterns*. Uma única assinatura de comportamentos pode identificar vários binários maliciosos. Estes mecanismos ajudam na detecção de *malware* que consiga gerar novas variantes (polimorfismo e metamorfismo), pois o *malware* neste caso irá sempre usar da mesma maneira os recursos e serviços do sistema. Um detector baseado em comportamentos contém tipicamente os seguintes componentes:

- **Colecção de dados:** este componente colecciona os dados de comportamento dos programas ou sistemas.
- **Interpretação:** este componente converte a informação coleccionada pelo componente anterior em representações intermédias.
- **Algoritmo:** É usado para comparar as representações intermédias com assinaturas de comportamento.

3.3.2 Análise de *malware*

A análise de *malware* [3][6][10] está ligada ao conceito de virtualização [17]. Esta tarefa de análise de *malware* é bastante frequente quando o conceito de *honeypot* se junta ao de virtualização. Através do *honeypot* são capturados binários suspeitos de actividade maliciosa e depois é frequente analisar esses binários quanto ao seu comportamento. As empresas de anti-*malware* fazem este tipo de análise todos os dias com o objectivo específico de criar assinaturas para detectar novas variantes de *malware*. Esta análise pode ser realizada de forma estática, através da leitura de código, ou pode ser realizada de forma dinâmica, executando o binário malicioso em causa. Na análise estática a virtualização pode ser uma mais valia para efeitos de contenção de uma possível infecção em situações de execução acidental do binário malicioso.

No caso da análise dinâmica a virtualização ainda faz mais sentido e é mais útil dada a perigosa tarefa da execução real de *malware*. A ideia neste caso consiste em ter um ambiente protegido para execução dos binários maliciosos.

Para um *worm* existem duas variantes desta ideia, a primeira variante diz explicitamente que o ambiente deve representar uma única máquina e o comportamento do *worm* é observado dentro da máquina da vítima. A segunda ideia consiste num ambiente que representa uma rede com várias máquinas de modo a ser observado o modelo e forma de propagação do *worm*. Existem inúmeras vantagens inerentes ao uso da virtualização para construir ambientes que realizam análise dinâmica de *malware*, uma prende-se ao facto do *malware* ser realmente executado sem se aperceber de tal situação. Outra vantagem tem a ver com o facto do *malware* ser analisado num ambiente isolado de sistemas de produção. Depois garantem uma solução bastante flexível e eficiente nomeadamente em termos dos recursos utilizados.

Em suma, neste projecto adoptou-se a técnica de análise dinâmica em que se executa o *malware* capturado (*worms* e *bots*) por *honeypots*, num ambiente controlado com o intuito de observação do comportamento dessas pragas virtuais. A observação de comportamento tem por base a monitorização de diversos artefactos de infecção, ficheiros, chaves de registo, tráfego de rede, processos e objectos

de exclusão mútua criados.

3.4 Resumo

Neste capítulo foram apresentados os assuntos mais teóricos relacionados com o contexto do projecto, desde a descrição das várias categorias de *malware* realçando em mais detalhe as categorias abrangidas pelo projecto, os *worms* e os *bots*. Foi dado um exemplo real de dois *worms* que ainda estão bastante activos na *Internet*, o *Conficker* e o *Stuxnet*, assim como, um exemplo real de um *worm* que tem comportamento de *bot*, o *Kolab*. Tanto o *Conficker* como o *Kolab* estiveram em grande foco na avaliação experimental deste projecto como poderá será visto mais à frente no capítulo 6. Foi apresentada a teoria inerente ao conceito de *honeypot/honeynet* assim como a referência introdutória aos processos de identificação e análise de *malware*.

No próximo capítulo os grandes temas apresentados neste capítulo (captura, identificação e análise de *malware*) serão instanciados através das plataformas de *software* seleccionadas para integrar este projecto.

Capítulo 4

Plataformas utilizadas

Neste projecto foi feita uma investigação sobre algumas plataformas de *software* que permitem realizar as tarefas de captura, identificação e análise dinâmica de *malware* que se propaga automáticamente.

O *dionaea* foi o *honeypot* escolhido para realizar a tarefa de captura de tráfego supostamente malicioso, o *VirusTotal* para realizar a tarefa de identificação ou detecção dos binários suspeitos de actividade maliciosa e por fim o *Anubis* para analisar dinamicamente esses binários.

A discussão das alternativas e a razão destas escolhas estão implicitamente explicadas neste capítulo e explicitamente descritas a pormenor no capítulo 7.

4.1 *Dionaea*

O *dionaea* é um *honeypot* de média interactividade, e é sobretudo considerado o verdadeiro sucessor do *nepenthes* assim como também é considerado por muitos especialistas de segurança como o melhor *honeypot* da actualidade.

Este *honeypot* foi desenvolvido como parte de um projecto para o *Honeynet/-Google Summer of Code 2009*[1]. O seu principal objectivo consiste em atrair tráfego malicioso/*malware* através de um conjunto de serviços vulneráveis e desta forma conseguir obter o binário malicioso. Actualmente o projecto *dionaea* conta com o apoio da *SURFnet*[2].

4.1.1 Características

- Escrito em *C* e *Python*[3].
- Suporte para *IPv6* assim como para o *Transport Layer Security (TLS)*, usa a

[1] `http://honeynet.org/gsoc2009/`
[2] `http://www.surfnet.nl/en/`
[3] `http://python.org/`

libemu[4] para detecção de *shellcode* e emulação da execução de binários supostamente maliciosos.

- Inclui o protocolo *Windows Server Message Block (SMB)* escrito em *Python*. É importante salientar que a grande maioria das vulnerabilidades de segurança do *Microsoft Windows* situam-se nos serviços disponibilizados por este protocolo.

- Integração com a ferramenta *p0f*[5] para *fingerprinting* passivo do sistema operativo da fonte de ataque.

- Log directo (pré-agregação da informação de tráfego capturado) para uma base de dados *SQLite*[6] através do seu sistema de comunicação interno (*incident/ihandler*).

- Possibilidade de integração com o *Extensible Messaging and Presence Protocol (XMPP)*[7] para envio/recepção de notificações e binários em tempo real.

- Fácil produção de gráficos estatísticos com o *gnuplot*[8] através dos dados presentes na base de dados *SQLite*.

4.1.2 Arquitectura

O *dionaea* possui uma arquitectura muito bem estruturada mas com bastantes sub-componentes, na figura 4.1 é feito um resumo esquemático dessa arquitectura.

4.1.3 Configuração

O *dionaea* permite através do ficheiro *dionaea.conf* uma configuração personalizada de acordo com as necessidades do utilizador. Existem várias secções com opções e parâmetros de configuração disponíveis no ficheiro *dionaea.conf*, porém nem todas as secções e opções de configuração têm de ser definidas para que o *honeypot* funcione. Dentro das secções de configuração mais importantes temos:

- **Secção de *Logging***
  Permite alterar os níveis de *log* do *dionaea* (*debug*, *warning*, etc).

[4] `http://libemu.carnivore.it/`
[5] `http://lcamtuf.coredump.cx/p0f.shtml`
[6] `http://sqlite.org/`
[7] `http://xmpp.org/`
[8] `http://gnuplot.info/`

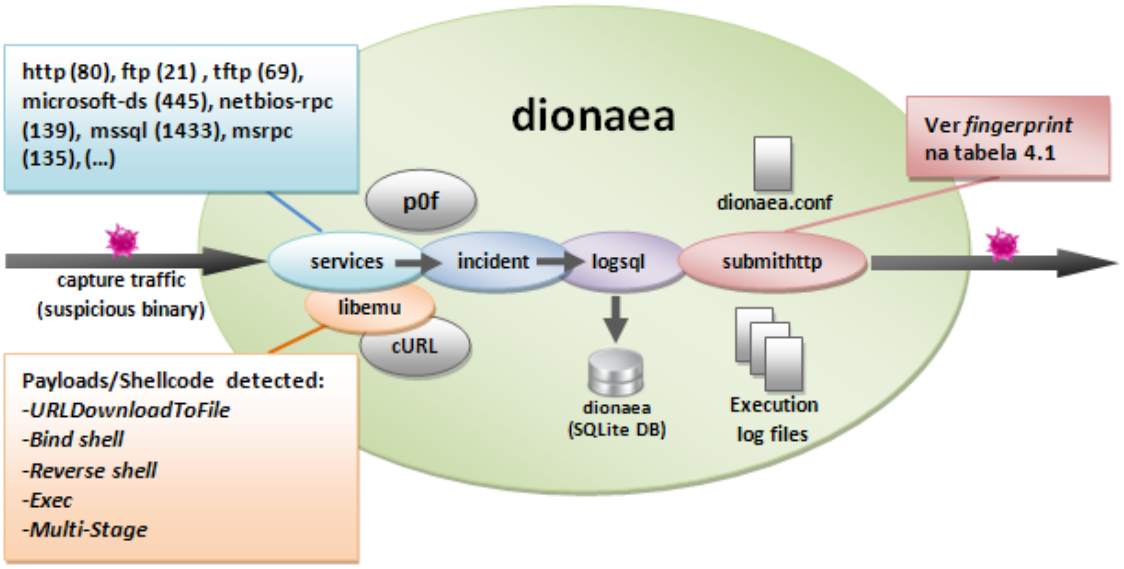


Figura 4.1: *dionaea*

- **Secção de *IP***
  Por defeito o *dionaea* faz *bind* automaticamente a todas as interfaces de rede disponíveis e respectivos endereços *IP*, tanto em *IPv4* como *IPv6*. Existe ainda a possibilidade de configuração manual das interfaces de rede e respectivos endereços *IP* para associação ao *dionaea*.

- **Secção de Módulos**
  Nesta secção pode-se activar, desactivar, e configurar várias funcionalidades e ferramentas usadas pelo *dionaea*. Em particular destacam-se as sub-secções, *ihandlers* e *services*.

4.1.4 Conectividade

O *dionaea* pode oferecer serviços via *Transmission Control Protocol (TCP)* e/ou *User Datagram Protocol (UDP)* sobre *IPv4* ou *IPv6*.

4.1.5 Protocolos

Para permitir ajustes fáceis aos protocolos foi utilizada a linguagem *Python*, e para as operações de mais baixo nível foi utilizada a linguagem *C*.
O *dionaea* inclui implementações dos protocolos *SMB*, *HTTP (Hypertext Transfer Protocol)*, *FTP (File Transfer Protocol)*, *TFTP (Trivial File Transfer Protocol)*, *TDS (Tabular Data Stream)* que é usado pelo *Microsoft SQL Server* e inclui ainda uma implementação do *SIP (Session Initiation Protocol)* para *VoIP (Voice over IP)*.

Estas implementações de protocolos disponibilizam serviços propositadamente vulneráveis (ver figura 4.1).

4.1.6 Funcionamento

Exploitation

O *dionaea* tem a capacidade para detectar e avaliar *shellcode* [20] presente no *payload* dos *exploits* [20] com o objectivo principal de ganhar uma cópia do binário malicioso.
Chama-se *shellcode* [13][20] a um conjunto de instruções usadas como *payload* do *exploit*, é tipicamente escrito em *assembly* e a ideia principal consiste em comprometer a máquina vulnerável obtendo a sua *shell* com privilégios de *Administrator* ou de *root*, depois desta ter executado as instruções que permitem tal feito.
Em vários casos o *dionaea* pode mesmo detectar *shellcode* e se necessário executá-lo virtualmente na *libemu*. A detecção do *shellcode* é realizada recorrendo a heurísticas avançadas. A explicação dessas heurísticas sai fora do contexto do projecto, no entanto é importante referir que dado o facto da detecção de *shellcode* ser uma tarefa bastante pesada computacionalmente, todo o processo realizado pela *libemu* na detecção de *shellcode* recorre a um mecanismo avançado de *multithreading*.
A engenharia reversa do *shellcode* é feita através da sua execução na *libemu vm* (*virtual machine*) e guardando as respectivas chamadas às funções da *API* do *Windows* assim como os argumentos dessas funções.
Para a grande maioria do *shellcode* o registo das chamadas à *API* do *Windows* é suficiente para se ter uma ideia da intenção do ataque, deste modo o *dionaea* (em conjunto com a *libemu*) constrói um pequeno *profile* de execução do *shellcode* e consegue desta forma perceber os passos do ataque.

Payload/Shellcode

Uma vez que o *dionaea* consiga obter o *payload/shellcode* do *exploit* e tenha construído com sucesso o respectivo *profile* de execução, tenta agir de acordo com a sua previsão.
Neste momento o *dionaea* em conjunto com a *libemu* pode detectar e construir *profiles* de execução para os seguintes tipos de *payload/shellcode*:

- ***URLDownloadToFile***
  Este tipo de *payload/shellcode* faz uma chamada à função *URLDownloadToFile()* da *API* do *Windows*, para que seja feito *download* de um ficheiro malicioso

via *HTTP* e de seguida executá-lo.

- ***Bind Shell***
  Este tipo de *payload/shellcode* (ver figura 4.2) associa (*bind*) uma *shell* a uma determinada porta, ficando à escuta (*listening*) pela ligação do atacante. Tipicamente após o atacante conseguir ligação à máquina da vítima, envia-

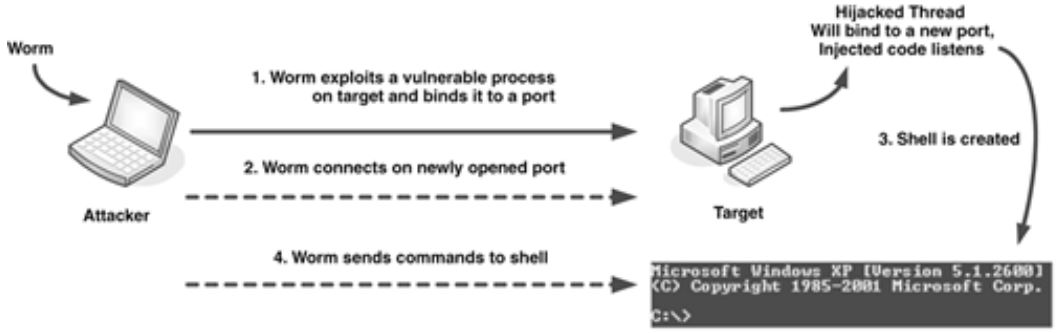


Figura 4.2: *Bind shell payload*

lhe instruções para esta fazer *download* de um ficheiro malicioso via *FTP* ou *TFTP*.

- ***Reverse Shell***
  Este tipo de *payload/shellcode* (ver figura 4.3) é bastante eficiente na evasão de *firewalls* e de mecanismos de *NAT (Network Address Translation)*, pois ao contrário do *Bind Shell*, aqui é a máquina comprometida que se liga ao atacante oferecendo-lhe a sua *shell*.

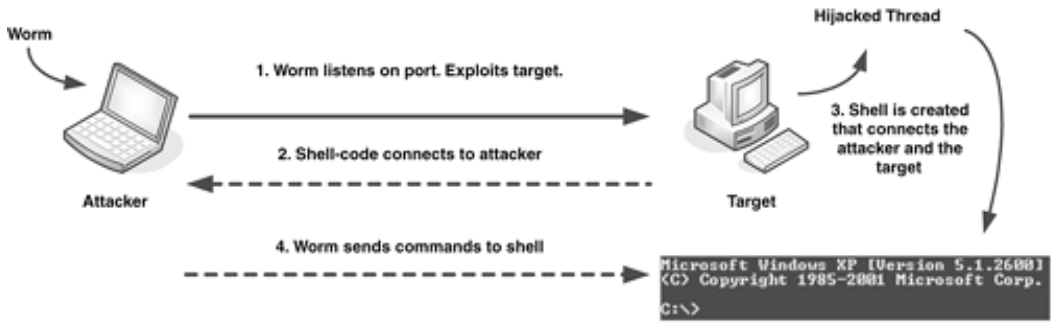


Figura 4.3: *Reverse shell payload*

- ***Exec***
  Neste tipo de *payload/shellcode* (ver figura 4.4) é chamada a função *WinExec()* da *API* do *Windows* com o objectivo de executar um ficheiro na máquina comprometida.

Figura 4.4: *Exec payload*

- ***Multi-Stage***
  Este tipo de *payload/shellcode* é composto por duas fases. Neste caso a *libemu vm* é usada para executar o *payload/shellcode* da primeira fase para conseguir prever as intenções do *payload/shellcode* da segunda fase.

Downloads

Depois do *dionaea* conhecer o *Uniform Resource Locator (URL)* para *download* do binário supostamente malicioso (por exemplo após execução do *URLDownload-ToFile payload*) trata imediatamente da sua transferência a partir do respectivo *URL*. Os protocolos para *download* de ficheiros via *FTP* e *TFTP* são implementados em *Python* (*ftp.py* e *tftp.py*). Para fazer *download* de ficheiros via *HTTP* é utilizado o *cURL*[9] que faz uso da *libcurl*[10].

Submit

Uma vez que o *dionaea* tenha conseguido obter uma cópia do binário supostamente malicioso é de seguida enviada a informação relativa ao ataque em que está envolvido o binário em causa (ver tabela 4.1). No caso deste projecto, o binário só é submetido se este ainda não existir no repositório de binários do servidor central. Todo o processo de submissão de dados da sonda de rede (onde o *dionaea* está em execução) para o servidor central é realizado através de uma versão modificada do sub-componente *submithttp* original do *dionaea* que por sua vez utiliza o *cURL* para envio dos dados por *HTTP POST*[11].

Fingerprint do evento de ataque

- ***Collector***: Endereço *IP* da sonda que inclui o *dionaea honeypot*.

[9] `http://curl.haxx.se/`
[10] `http://curl.haxx.se/libcurl/`
[11] `http://en.wikipedia.org/wiki/POST_(HTTP)`

Collector	MD5	SHA512	SourceHost	SourcePort	TargetHost	TargetPort	Service	DownloadURL	Submitted

Tabela 4.1: *Fingerprint* do evento de ataque enviada a partir da sonda para o servidor central.

- ***MD5***: *Message-Digest 5 Algorithm (MD5)*[12] do binário supostamente malicioso.
- ***SHA512***: *Secure Hash Algorithm (SHA512)*[13] do binário supostamente malicioso.
- ***SourceHost***: Endereço *IP* da fonte de ataque.
- ***SourcePort***: Porta da fonte de ataque.
- ***TargetHost***: Endereço *IP* do alvo de ataque (*dionaea honeypot IP*).
- ***TargetPort***: Porta atacada.
- ***Service***: Serviço atacado.
- ***DownloadURL***: *Uniform Resource Locator (URL)* para download do binário supostamente malicioso.
- ***Submitted***: Data e hora de ataque (formato *time zone*).

logsql handler

O sub-componente *logsql handler* trata do processo de pré-agregação e escrita de informações de ataque (incluindo a informação capturada pelo componente *p0f*) para uma base de dados *SQLite* embebida na própria sonda.

4.2 *VirusTotal*

O *VirusTotal* (ver figuras 4.5 e 4.6) é um anti-*malware* remoto e gratuito mantido pela empresa espanhola *Hispasec Sistemas*[14].
Este serviço remoto tem a grande vantagem de actualizar com bastante frequência as suas assinaturas de *malware* e é considerado por muitos o melhor e mais robusto anti-*malware* remoto no que diz respeito à identificação/detecção de ficheiros

[12] `http://en.wikipedia.org/wiki/MD5`
[13] `http://en.wikipedia.org/wiki/SHA-2`
[14] `http://hispasec.com/`

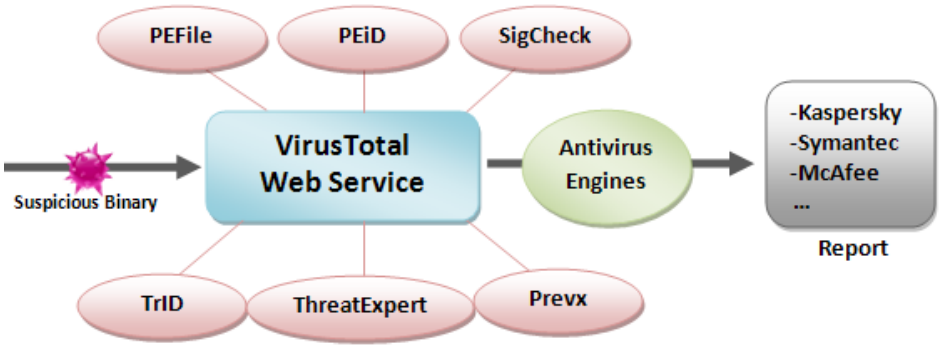
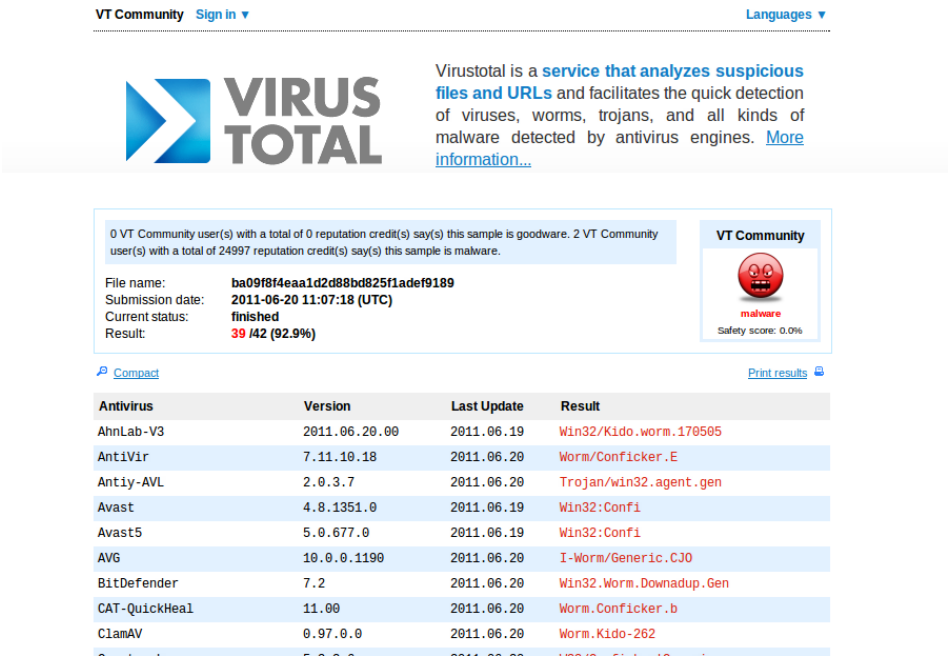


Figura 4.5: *VirusTotal*

suspeitos.
Dispõe de uma plataforma (*web service*) que permite fazer *upload* de ficheiros suspeitos até 20 MB e realiza uma tentativa de identificação em mais de 40 motores de anti-*malware* diferentes. Como resultado de identificação/detecção de ficheiros suspeitos são devolvidos valores como:


VT Community Sign in ▾ Languages ▾

VIRUS TOTAL

Virustotal is a **service that analyzes suspicious files and URLs** and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. More information...

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 2 VT Community user(s) with a total of 24997 reputation credit(s) say(s) this sample is malware.

File name: ba09f8f4eaa1d2d88bd825f1adef9189
Submission date: 2011-06-20 11:07:18 (UTC)
Current status: finished
Result: 39 /42 (92.9%)

VT Community
malware
Safety score: 0.0%

Compact Print results

Antivirus	Version	Last Update	Result
AhnLab-V3	2011.06.20.00	2011.06.19	Win32/Kido.worm.170505
AntiVir	7.11.10.18	2011.06.20	Worm/Conficker.E
Antiy-AVL	2.0.3.7	2011.06.20	Trojan/win32.agent.gen
Avast	4.8.1351.0	2011.06.19	Win32:Confi
Avast5	5.0.677.0	2011.06.19	Win32:Confi
AVG	10.0.0.1190	2011.06.20	I-Worm/Generic.CJO
BitDefender	7.2	2011.06.20	Win32.Worm.Downadup.Gen
CAT-QuickHeal	11.00	2011.06.20	Worm.Conficker.b
ClamAV	0.97.0.0	2011.06.20	Worm.Kido-262

Figura 4.6: *VirusTotal Web Report*

- *MD5* do ficheiro.
- Descrição da identificação/detecção do ficheiro em cada motor de anti-*malware* (*Kaspersky*, *Symantec*, *McAfee*[15], etc).

[15] `http://mcafee.com/`

- Tipo de ficheiro (através da ferramenta *TrID*[16]).
- *Timestamp* do ficheiro.
- *Imports* e *exports* realizados pelo ficheiro (através da ferramenta *PEFile*[17]).
- Informação acerca da validade da assinatura digital do ficheiro (através da ferramenta *SigCheck*[18]).
- Nome dos *packers* usados para camuflar o ficheiro (através da ferramenta *PEiD*[19]).
- Pequena descrição do ficheiro se o seu *MD5* for encontrado na *National Software Reference Library (NSRL)*[20].
- Link da análise realizada pelo serviço *Prevx*[21].
- Link da análise realizada pelo serviço *ThreatExpert*[22].

4.3 *Anubis*

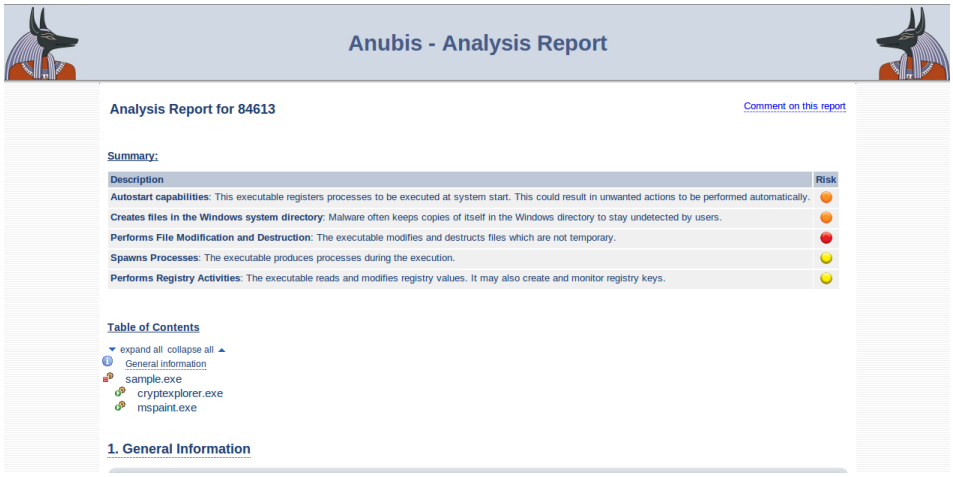


Figura 4.7: *Anubis Web Report*

O *Anubis* (ver figura 4.7) [2][3][4][5][6] é uma plataforma remota criada na TU Vienna[23] que fornece serviços gratuitos de análise dinâmica de executáveis *Windows-PE (Portable Executable)*, com foco especial na análise dinâmica de *malware*.

[16] http://mark0.net/soft-trid-e.html
[17] http://code.google.com/p/pefile/
[18] http://technet.microsoft.com/en-us/sysinternals/bb897441/
[19] http://peid.info/
[20] http://www.nsrl.nist.gov/
[21] http://prevx.com/
[22] http://threatexpert.com/
[23] http://tuwien.ac.at/EN/

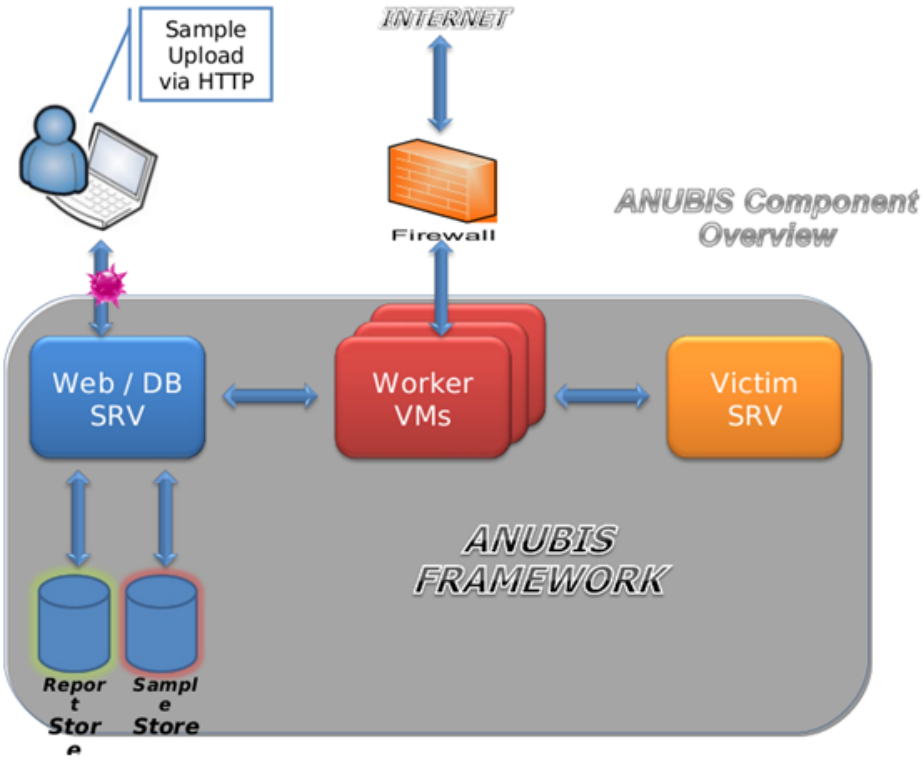


Figura 4.8: *Anubis*

Nesta plataforma os binários são sempre executados num sistema computacional completamente emulado, como por exemplo a *Central Processor Unit (CPU)*. Neste contexto a ferramenta *Qemu*[24] é usada para a emulação do ambiente computacional e o sistema operativo anfitrião é o *Microsoft Windows XP Service Pack 2 (SP2)*[25] emulado e completamente transparente para o binário.
Todas as acções de execução são monitorizadas, como por exemplo, as chamadas às funções da *API* do *Windows*.

4.3.1 Arquitectura e componentes

Como se pode confirmar na figura 4.8, o *Anubis* tem 5 blocos primários ou componentes de construção, são eles, *Web/DB Server*, *Malware Sample Storage*, *Report Storage*, *Victim Server* e *Worker (VM) Images*.

- ***Web/DB Server***
  Servidor *web* que permite receber os binários suspeitos (referido como *sample* na figura 4.8) para análise dinâmica.
  Armazena as referências para os binários numa base de dados relacional assim como os respectivos resultados de análise dinâmica.

[24] `http://wiki.qemu.org/`
[25] `http://windows.microsoft.com/`

- ***Malware Sample Storage***
  Responsável por armazenar os binários suspeitos.

- ***Report Storage***
  Responsável por armazenar os relatórios/ficheiros com o resultado da análise dinâmica dos binários suspeitos.

- ***Victim Server***
  Funciona como um *honeypot* local para certos serviços.

- ***Worker (VM) Images***
  Realizam todo o trabalho de análise dinâmica.

4.3.2 Ferramentas utilizadas

O *Anubis* para realizar a análise dinâmica de binários recorre a vários componentes de *software* desenvolvidos por terceiros:

- ***Qemu***
  É uma ferramenta livre que implementa um emulador de processador, permitindo uma virtualização completa de um PC. A *Qemu* é semelhante a ferramentas como a *VMware Workstation*[26] e o *PearPC*[27], mas inclui uma vantagem fulcral em relação aos últimos, a especialização para arquitecturas *x86*.

- ***PEiD***
  Detecta a maioria dos *packers*, *cryptors* e *compilers* mais comuns para ficheiros *Windows-PE*. Pode detectar actualmente mais de 600 assinaturas de ficheiros *Windows-PE* diferentes.

- ***BRO***[28]
  É uma ferramenta *open-source* baseada no *Unix Network Intrusion Detection System* (*NIDS*) que monitoriza passivamente o tráfego de rede e procura por actividades suspeitas.

- ***Sigbuster***[29]
  É uma ferramenta *cross-plataform* para detecção de *packers-exe*.

[26] `http://vmware.com/products/workstation/`
[27] `http://pearpc.sourceforge.net/`
[28] `http://bro-ids.org/`
[29] `http://teamfurry.com/`

4.3.3 Resultados de análise dinâmica

O *Anubis* monitoriza várias actividades antes, durante e após a execução de um binário suspeito nas suas *Worker (VM) Images*.
Actualmente é realizada a monitorização e consequente criação de relatórios (em *XML, PDF* ou *HTML*) para os cinco seguintes grupos de actividades possíveis:

- **Actividades no sistema de ficheiros**
  Leituras/escritas em ficheiros, criação de novos ficheiros.

- **Actividades no *Windows Registry***
  Chaves de registo criadas/removidas/alteradas.

- **Actividades nos processos**
  Processos criados/terminados/que comunicaram com outro processo.

- **Actividades nos serviços do *Windows***
  Serviços que foram iniciados/terminados.

- **Actividades de rede**
  Consultas de *DNS*, downloads por *HTTP/FTP*, sessões de conversação via *SMTP/IRC* (importante para a detecção de *bots/botnets*).

4.4 Projectos relacionados

4.4.1 *Leurre.com*

O projecto *Leurre.com*[30] envolve a investigação de vários especialistas da *Symantec* e é baseado no *Fabien Pouget's Honeynet Project (v1.0)*[31] e numa infra-estrutura distribuída para tratar de *zero-day exploits*, a *SGNET*.

A *SGNET* é constituída por um misto de tecnologias, o *Scriptgen (Eurecom)*[32], *Argos (VU Amsterdam)*[33], *Nepenthes* (*TU Mannheim*[34]), *Anubis (TU Vienna)* e o *Virustotal (Hispasec Sistemas)*.

O projecto envolve cerca de 200 *honeypots* espalhados pelo mundo que capturam *malware* automaticamente, e posteriormente com ajuda das tecnologias que integra, realiza uma análise aos binários maliciosos capturados.

[30] `http://www.leurrecom.org/`
[31] `http://eurecom.fr/people/pouget.en.htm`
[32] `http://www.eurecom.fr/index.en.htm`
[33] `http://vu.nl/en/index.asp`
[34] `http://www.uni-mannheim.de/1/english/startpage/index.html`

4.4.2 *WOMBAT*

Este projecto Europeu[35] foi iniciado em 2008 e tem como grandes parceiros a *VU Amsterdam*, *Eurecom*, *FORTH*, *PoliMilano*[36] e *TU Vienna*. Basicamente trata-se de um observatório do comportamento das ameaças de ataque na *Internet*.

O objectivo principal do *WOMBAT* é encontrar novas formas de compreensão do comportamento das ameaças emergentes na *Internet*, e para tal implementa a automatização da análise de *malware* através da plataforma *Anubis*.

Tanto o projecto *Leurre.com* como o *WOMBAT* pecam por incluir algumas tecnologias obsoletas, como por exemplo o *nepenthes honeypot* para captura de tráfego suspeito de actividade maliciosa.

4.5 Resumo

Neste capítulo foram referidos e detalhados pormenores de construção e funcionamento de cada uma das plataformas de *software* de terceiros que foi usada na construção do *WMS*. Pode-se afirmar que foram instanciadas as plataformas de *software* para três grandes temas apresentados teoricamente no capítulo 3, o *dionaea honeypot* para a captura de binários supostamente maliciosos envolvidos em ataques, o *VirusTotal* para identificação dos binários, e o *Anubis* para efeitos de análise dinâmica. Foram ainda apresentados muito resumidamente, dois grandes projectos, o *Leurre.com* e o *WOMBAT* que são parcialmente similares no que diz respeito às ideias e temáticas abordadas neste projecto.

No próximo capítulo as plataformas de *software* de terceiros são referidas para efeitos de explicação de todo o mecanismo de funcionamento do *WMS*, assim como é detalhado todo o mecanismo de integração destas plataformas na sua construção.

[35] `http://wombat-project.eu/`
[36] `http://www.english.polimi.it/`

Capítulo 5

WMS

Para testar e validar as ideias incluídas neste documento a nível funcional e estatístico foi idealizado, desenhado e construído o *WMS*. Foram consideradas com o maior rigor possível as boas práticas e algumas recomendações de segurança propostas pelos projectos *Honeynet*, *Carnivore*[1] e *Kaspersky Labs*.
Na realização deste projecto foi considerada a possibilidade do sistema desenvolvido poder ser integrado como solução final, em ambiente de produção, no sistema *Pulso* da PT. Desta forma foram requeridos determinados padrões arquitecturais assim como requisitos de metodologia de implementação levados a cabo no desenvolvimento do *WMS* e dos seus componentes constituintes.

Neste capítulo é abordado o desenvolvimento do projecto *WMS* assim como detalhes de funcionamento dos seus componentes internos ou aplicações adicionais para um dos componentes internos do *WMS*, o *mwmonitor*.

5.1 Aproximação

O *WMS* foi concebido especificamente para automatizar o processo de captura, identificação/detecção e análise automática de *malware* que possa eventualmente ser propagado em grandes redes empresarias, sendo que os *worms* e os *bots* são neste contexto os exemplos mais sonantes. Desta forma a equipa operacional de segurança poderá através do *WMS* identificar e resolver rapidamente os respectivos incidentes de ataque via *malware*.

A arquitectura deste sistema constitui um servidor central e várias sondas de rede (da plataforma *Pulso*) estrategicamente distribuídas pela rede empresarial, estas não comunicam entre si mas comunicam directamente com o servidor central. Uma sonda de rede (capítulo 2) consiste neste caso num computador com recursos computacionais reduzidos e no contexto empresarial tipicamente não dispõe de conectividade para o exterior, apenas interior (*Intranet*). O servidor

[1] `http://carnivore.it/`

central consiste num computador com recursos computacionais mais elevados que os recursos computacionais das sondas e dispõe de conectividade para o exterior (*Internet*).

Para efeitos de desenvolvimento e experimentação do projecto, o *WMS* foi construído tendo por base a grande rede empresarial da PT. Numa primeira fase foram instalados os componentes constituintes e respectiva execução nos sistemas computacionais dentro dessa rede. Na segunda fase os componentes constituintes foram instalados e executados em sistemas computacionais fora da rede empresarial dando conectividade para o exterior (*Internet*) às sondas. O objectivo da segunda fase passou por verificar com mais exactidão o funcionamento e/ou comportamento do *WMS* no processo de captura, identificação/detecção e análise dinâmica de binários suspeitos assim como no processo da respectiva produção de alarmes e estatísticas face a tráfego intenso e consequentemente perante a ocorrência de ataques reais.

5.2 Arquitectura

Como se pode comprovar pela figura 5.1 o processo de captura e pré-agregação de informação do tráfego possivelmente malicioso, incluindo binários maliciosos que possam eventualmente estar envolvidos no ataque, é realizado por dois componentes e dois sub-componentes no lado da sonda, o *dionaea honeypot* para capturar tráfego malicioso (incluindo os binários supostamente maliciosos) que controla o sub-componente *logsql handler*. O *logsql handler* por sua vez permite guardar a informação sobre o tráfego capturado (para uma pequena base de dados *SQLite*) assim como a informação conseguida por um segundo componente independente, o *p0f*, que por sua vez permite realizar *fingerprinting* passivo do sistema operativo da fonte de ataque. A informação capturada pelo *p0f* chega ao *dionaea* através do sub-componente *p0f handler* e de um *socket* local. O envio de informação relevante sobre ocorrências de ataque assim como os binários maliciosos inerentes a cada ataque são enviados através do sub-componente *submithttp* para o servidor central.

No servidor central os dados enviados pela sonda, ou seja, a informação sobre as ocorrências de ataque que envolvam binários supostamente maliciosos é agregada e guardada numa base de dados através do componente *receivehttp*. Como será visto mais à frente este componente tem ainda a responsabilidade de definir o estado do evento de ataque que será importante para os processos de identificação, análise e alarmística. Para controlo dos sub-componentes que estão responsáveis pelo processo de identificação/detecção automática dos binários suspeitos de actividade maliciosa e respectiva análise dinâmica recorre-

se ao componente *mwmonitor* no lado do servidor central.
O processo de identificação/detecção de *malware* é controlado principalmente pelo sub-componente *avsubmit* em conjunto com o anti-*malware* remoto *VirusTotal* enquanto que o processo de análise dinâmica é controlado pelo sub-componente *ansubmit* em conjunto com a plataforma remota *Anubis*.

Todo o processo deverá ser monitorizado por administradores de segurança responsáveis pela *QoP* da rede empresarial e dos serviços que esta disponibiliza. Em paralelo e como o objectivo principal passa por resolver o incidente o mais rápido possível, a equipa técnica presente no *Secure Operations Center* (*SOC*) que à partida é especializada na resolução deste tipo de incidentes, deverá ser alertada acerca da situação em tempo quase real.

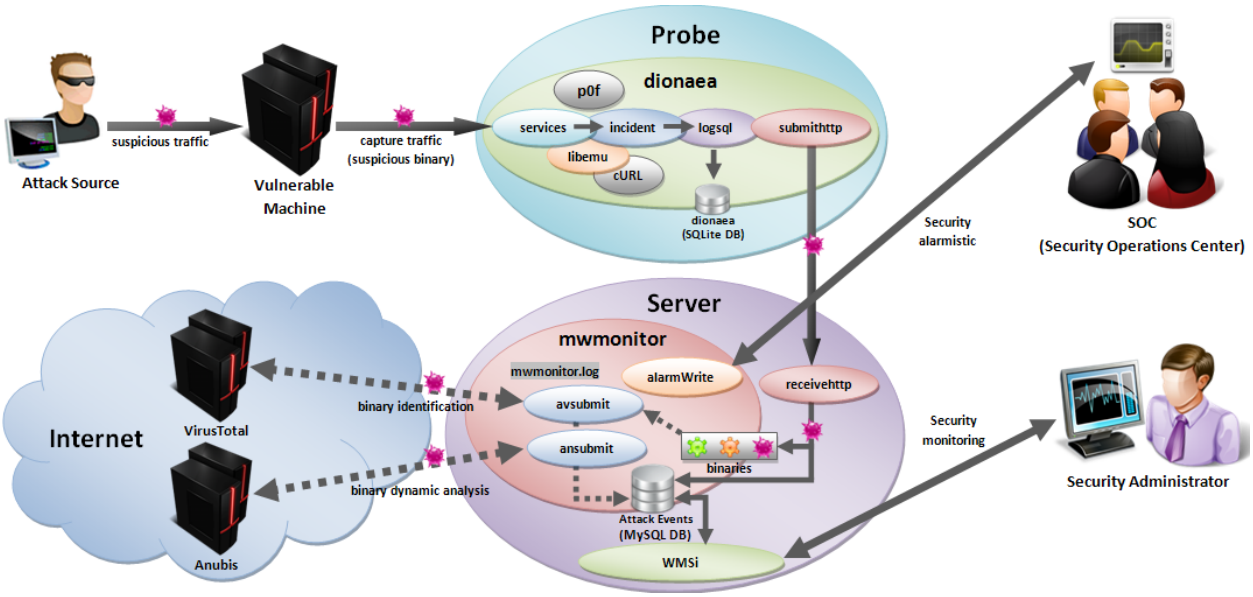


Figura 5.1: *WMS*

5.3 Componentes internos

5.3.1 *receivehttp*

Este componente (ver figura 5.2) foi escrito em *PHP* e encontra-se no servidor central sob controlo de um *Web Application Server*, o *Apache*[2]. Tem como principais objectivos controlar a recepção via *HTTP* e salvaguarda dos dados do evento de ataque que ocorreu na rede e que foi capturado no lado da sonda de rede pelo *dionaea*. Tem ainda a responsabilidade adicional de definir/classificar o estado desse evento de ataque.

[2] `http://apache.org/`

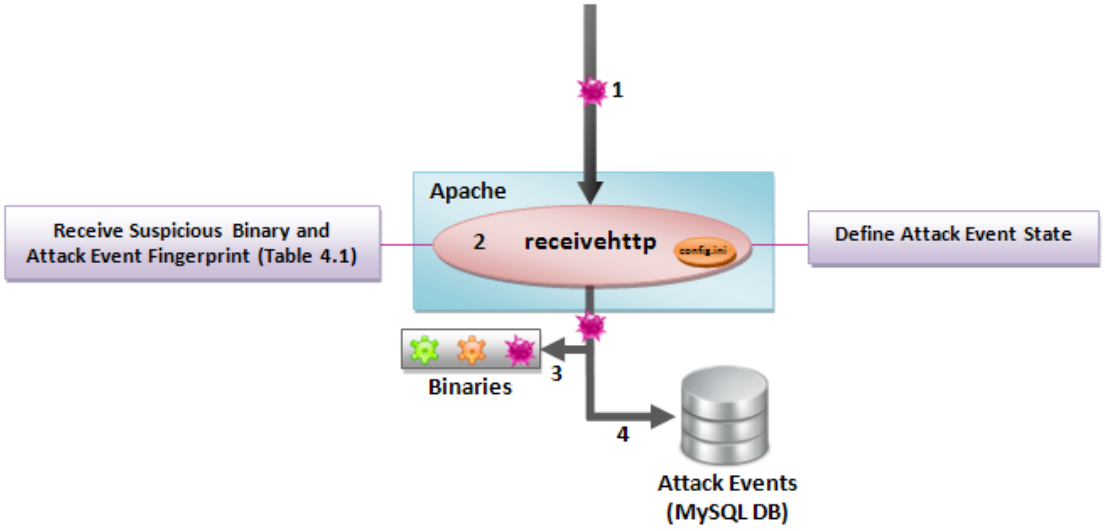


Figura 5.2: *receivehttp*

Funcionamento

1. **Recepção dos dados relativos ao evento de ataque ocorrido.**
   Os dados relativos ao evento de ataque enviados pela sonda (ver tabela 4.1) são recebidos pelo componente *receivehttp* no servidor central via protocolo *HTTP*.

2. **Definição e classificação do estado do evento de ataque ocorrido.**
   O estado do evento de ataque é definido e classificado de acordo com o número de ocorrências iguais (*i.e.* o mesmo *SourceHost*, *TargetPort* e *MD5*) já existentes na base de dados. Esta definição e classificação do estado é importante para evidenciar os eventos de ataque mais persistentes na rede, ter conhecimento da resolução ou não resolução da situação assim como fornecer a base para um processo de alarmística bem controlado, correcto e o mais eficiente possível. O mesmo estado é definido e classificado de acordo com a igualdade dos dados *SourceHost*, *TargetPort* e *MD5* já existentes noutros eventos guardados na base de dados central.
   Para um dado evento de ataque (x) recebido pelo servidor central, com estado (e) definido internamente por omissão como *NULL* ($e(x) = NULL$), podemos ter ou não um determinado número de ocorrências iguais ($\#o$) na base de dados central para x ($\#o(x) = [0..+oo[$).

```
SE (#o(x) >= 10) ENTÃO
   e(x) = EXTREME_PERSISTENCE
SENÃO SE (1 <= #o(x) < 10) ENTÃO
   e(x) = HIGH_PERSISTENCE
SENÃO
   e(x) = NOT_ALARM
```

Este algoritmo fornece a base para um processo de alarmística mais correcto e eficiente. Garante-se deste modo que não se volta a produzir o mesmo alarme para um dado evento de ocorrência de ataque que já existe na base de dados e que pode já ter o seu estado definido como *ALARM* (*e(x) = ALARM*), ou seja, pode já ter sido produzido um alarme para este evento de ocorrência de ataque.

Pode-se assim afirmar que cada evento de ataque único, isto é, cada evento de ataque que tenha o conjunto de dados *MD5, SourceHost* e *TargetPort* único, é candidado a gerar apenas um alarme, pois o seu estado nesta situação será sempre *NOT_ALARM* (*e(x) = NOT_ALARM*).

Se um evento de ataque fôr único apenas pode:

- Com estado *NULL* transitar para estado *NOT_ALARM*.
- Com estado *NOT_ALARM* transitar para estado *ALARM* (este processo não é da responsabilidade do componente *receivehttp* mas sim do *mwmonitor*).

Se um evento de ataque não fôr único apenas pode:

- Com estado *NULL* transitar para estado *HIGH_PERSISTENCE* ou *EXTREME_PERSISTENCE*.

3. **Salvaguarda do binário supostamente malicioso que foi envolvido no ataque.**
   A salvaguarda do binário supostamente malicioso no servidor central só ocorre se este não existir no seu repositório local, só neste caso é que o binário é submetido pela sonda através do *submithttp* para o servidor central para que este o guarde através do componente *receivehttp*. A verificação é realizada pelo *receivehttp* através do *MD5* recebido nos dados do evento de ataque em comparação com o *MD5* dos binários existentes no repositório do servidor central.
   Sendo assim existem 2 tipos de mensagens principais que podem ser enviados para o *dionaea* a partir do *receivehttp*:

- ***FILE_REQUEST***
  No caso do binário (*MD5*) não existir no repositório local. Após recepção do binário, o *receivehttp* envia a mensagem *S_FILEOK* para o *dionaea* notificando que o binário foi recebido com sucesso no servidor central.

- ***FILE_KNOWN***
  No caso do binário (*MD5*) já existir no repositório local. Neste caso o binário não volta a ser submetido pelo *dionaea*.

4. **Salvaguarda dos dados relativos ao evento de ataque ocorrido.**
   Independentemente do estado definido para um evento de ocorrência de ataque, a sua escrita (4.1 + atributo *State* com estado do evento) é sempre realizada na base de dados central para efeitos de identificação/detecção, análise dinâmica e monitorização das ocorrências de ataque.

5.3.2 *mwmonitor*

Este é um dos componentes fulcrais do *WMS* e está escrito (assim como os seus sub-componentes) em *Python*.
O *mwmonitor* é responsável pela geração e envio de alarmes assim como pelo controlo de todos os sub-componentes responsáveis pela automatização do processo de identificação/detecção e análise dinâmica (em conjunto com as plataformas externas *VirusTotal* e *Anubis*) de binários supostamente maliciosos que estejam envolvidos nos ataques.

Aproximação

O *mwmonitor* é um componente que está responsável por outros sub-componentes e consequentemente por várias tarefas cruciais no lado do servidor central.
Optou-se por utilizar o *cron* para executar de *'x'* em *'x'* minutos o componente *mwmonitor* em detrimento da sua execução como um *daemon*. De *'x'* em *'x'* minutos o *mwmonitor* é executado, seleccionam-se os binários que ainda não foram identicados/detectados nem analisados e através do sub-componente *avsubmit* cada binário seleccionado é submetido ao *VirusTotal* e na volta são obtidos os resultados de identificação/detecção correspondente.
Após a recepção dos resultados de identificação/detecção remota de cada binário, é utilizado o sub-componente *ansubmit* para que seja realizada a análise dinâmica do respectivo binário através da plataforma remota *Anubis*.

Arquitectura

Como se pode verificar na figura 5.3, o componente *mwmonitor* controla vários sub-componentes internos que estão responsáveis pela comunicação com as plataformas externas auxiliares *VirusTotal* e *Anubis* assim como pelo processo de alarmística.

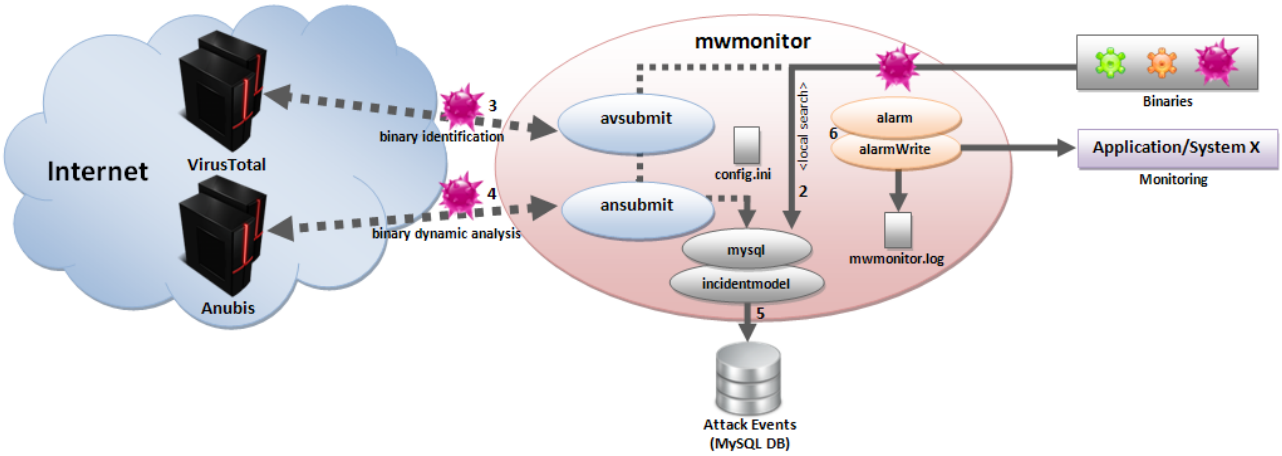


Figura 5.3: *mwmonitor*

Sub-componentes internos

- ***avsubmit***
  Este sub-componente está responsável por submeter o binário para o serviço remoto de anti-*malware VirusTotal* e obter os resultados de identificação/detecção a partir de vários motores de anti-*malware* diferentes.

- ***ansubmit***
  Este sub-componente está responsável por submeter o binário para a plataforma de análise dinâmica *Anubis* e por obter o *URL* para o respectivo resultado.

- ***mysql***
  Este sub-componente está responsável por controlar a ligação à base de dados central (*attackevents*).

- ***incidentmodel***
  Este sub-componente define as consultas a realizar na base de dados central.

- ***alarm***
  Este sub-componente permite criar os alarmes.

- ***alarmwrite***
  Este sub-componente está responsável por escrever/enviar alarmes.

- ***config.ini***
  Este ficheiro inclui os parâmetros de configuração para o componente *mwmonitor* e seus sub-componentes (*path* para a pasta local dos binários suspeitos, configurações para o *MySQL*, *Anubis*, etc).

Funcionamento

1. **Selecção de eventos de ataque para identificação/detecção e análise dinâmica**
   Inicialmente são escolhidos os eventos de ataque não identificados/detectados nem analisados a partir da base de dados central. O objectivo principal é identificar/detectar e analisar dinamicamente os binários supostamente maliciosos a partir dos respectivos *MD5* presentes em cada evento de ataque seleccionado.

2. **Verificação local da existência de identificação/detecção e análise**
   Como os eventos de ataque seleccionados podem ter o seu estado definido como *HIGH_PERSISTENCE* ou *EXTREME_PERSISTENCE* é verificado se o binário correspondente a cada um destes eventos já foi identificado/detectado e analisado numa sessão (*i.e.* execução do *mwmonitor*) anterior.
   Se já existir uma identificação/detecção e análise na base de dados central apenas é realizada uma cópia desta identificação/detecção para o novo evento de ataque recebido (cópia da identificação/detecção e análise para o respectivo binário associado), neste caso saltar para o passo 5, caso contrário continuar para o passo seguinte.

3. **Envio dos binários (*MD5*) para a plataforma de anti-*malware VirusTotal* e respectiva recepção dos resultados de identificação/detecção**
   O binário correspondente ao *MD5* de cada evento de ataque seleccionado começa por ser submetido para o anti-*malware* remoto *VirusTotal* com o subcomponente *avsubmit*. De seguida recebem-se os resultados de identificação no *avsubmit* que por sua vez retorna-os para o *mwmonitor* que os filtra para que apenas sejam aproveitados os resultados de identificação/detecção dos motores de anti-*malware* mais relevantes no mercado, são eles:

   - *Kaspersky*
   - *Symantec (Norton)*
   - *McAfee*
   - *NOD32*[3]
   - *Panda*[4]

[3] `http://eset.com/us/products/nod32.php`
[4] `http://pandasecurity.com/Antivirus`

- *Microsoft (MSE)*[5]
- *Avast*[6]
- *AVG*[7]
- *ClamAV*[8]

4. **Envio dos binários (MD5) para a plataforma de análise dinâmica *Anubis* e respectiva recepção do *URL* da página *web* com os resultados**
   O *mwmonitor* faz uma chamada ao *ansubmit* para que seja submetido o binário para a plataforma *Anubis* com o intuito principal de se obter o *URL* da página *web* que contém os resultados da análise dinâmica.

5. **Registo na base de dados central da informação de identificação/detecção e *URL* com resultados de análise dinâmica para os binários em causa**
   Após obtenção dos resultados de identificação/detecção e *URL* da análise dinâmica dos binários em causa a informação é armazenada na base de dados central. Os resultados são devidamente associados aos dados do respectivo evento de ataque que contém o *MD5* do binário em causa formando a chamada *fingerprint* completa. A *fingerprint* completa (ver tabela 5.1) inclui os resultados de identificação/detecção e *URL* de análise dinâmica do binário (*MD5*) envolvido no evento de ataque.

6. **Processo de alarmística**
   Sob controlo do componente *mwmonitor*, para cada evento de ataque classificado com estado *NOT_ALARM* é gerado um alarme (ver *fingerprint* completa em 5.1) através do sub-componente *alarm* e de seguida cada alarme é escrito/submetido para um ficheiro local (*mwalarm.log*) através do sub-componente *alarmwrite*.
   Se o alarme for escrito/submetido com sucesso, o estado do evento de ataque que gerou o alarme é actualizado para *ALARM*. Desta forma pode-se integrar facilmente todo o processo de alarmística com uma aplicação ou sistema independente, por exemplo enviando os alarmes gerados para a equipa operacional responsável pela resolução deste tipo de incidentes de segurança (*i.e.* equipa do *SOC*).

[5] `http://microsoft.com/pt-pt/security_essentials/`
[6] `http://avast.com/`
[7] `http://avg.com/`
[8] `http://clamav.net/`

Collector	MD5	SHA512	SourceHost	SourcePort	TargetHost	TargetPort	Service	DownloadURL	Submitted
State									
Kaspersky	Symantec	McAfee	NOD32	Panda	Microsoft	Avast	AVG	ClamAV	
Anubis									

Tabela 5.1: *Fingerprint* completa para alarmística.

Extensões ao *mwmonitor*

Uma das grandes vantagens do componente principal de *back-end* (o *mwmonitor*) que executa no servidor central tem a ver com o facto deste poder ser facilmente estendido para suportar novas funcionalidades que podem ser uma mais valia no contexto de um ambiente empresarial. Tendo tal facto em conta, foram idealizados e implementados dois protótipos (*mailrobot* e *artifactsrobot*) que usam as funcionalidades do componente *mwmonitor*.
Estes protótipos de extensão usufruem dos serviços controlados e disponibilizados pelo *mwmonitor* assim como usufruem da informação presente na base de dados do servidor central.

- ***mailrobot***

  O protótipo *mailrobot* foi escrito em *Python* e permite a recolha automática de binários suspeitos submetidos para um servidor de *mail* como *attachments*, com o objectivo principal de identificação/detecção e análise dinâmica.

  #### Arquitectura

  Como se pode verificar na figura 5.4 este protótipo é composto por um cliente (*btmclient*) que permite enviar binários suspeitos para um servidor de *mail* e por um componente (*mailbinaryrobot*) que executa no servidor central de *'x'* em *'x'* minutos. O *mailbinaryrobot* é auxiliado pelo *mwanalysis* para controlo e submissão dos binários suspeitos recolhidos do servidor de *mail* para as plataformas externas auxiliares *VirusTotal* e *Anubis*. Este último componente está ainda responsável por enviar *mails* de resposta com os resultados de identificação/detecção e análise dinâmica aos utilizadores que submeteram os binários suspeitos.

  #### Componentes

  – ***btmclient***
  Este componente funciona como o próprio nome indica, como um cliente

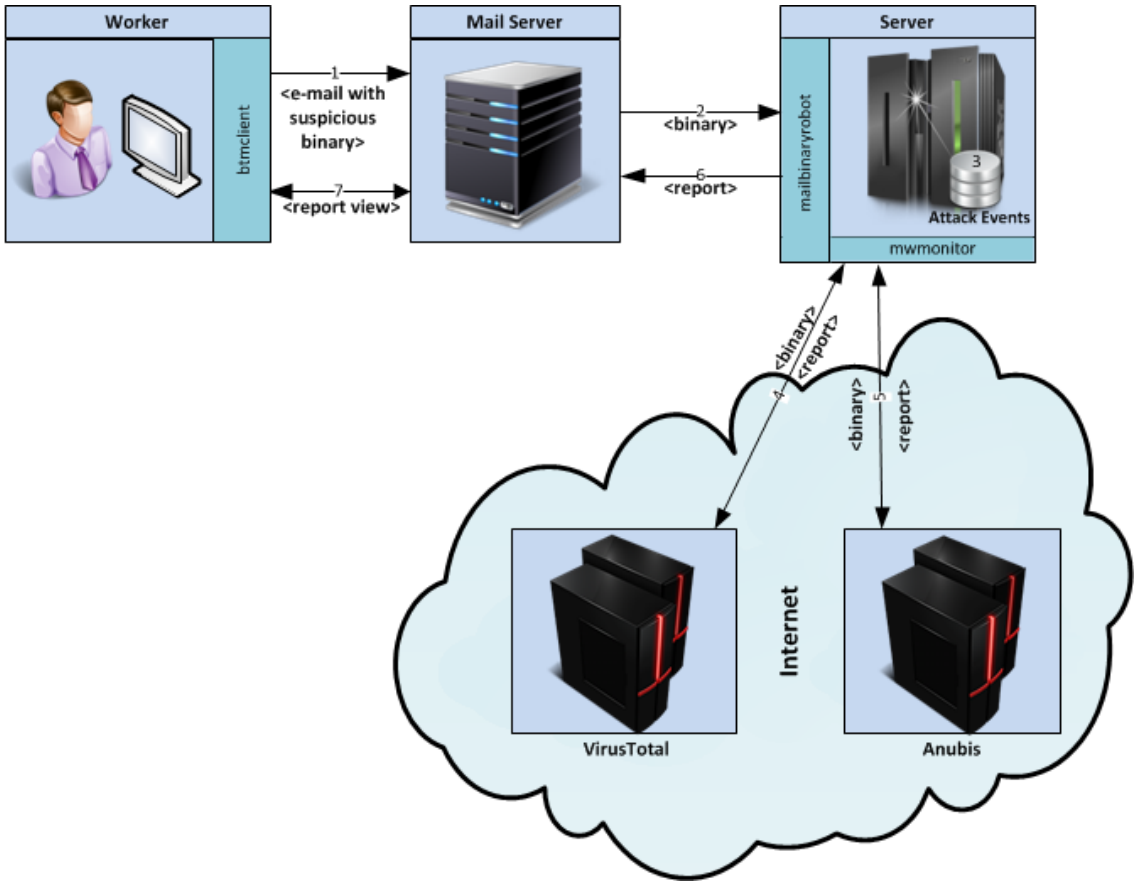


Figura 5.4: *mailrobot*

que permite enviar um *e-mail* (*i.e.* com um ficheiro binário anexado) para um servidor de *mail* através do *Simple Mail Transfer Protocol (SMTP)* e das *Multipurpose Internet Mail Extensions (MIME)*.

– ***btmclient.ini***
Este ficheiro contém dados de configuração (servidor e porta *SMTP*, localização dos binários suspeitos) para o *btmclient*.

– ***mailbinaryrobot***
Este componente encontra-se no servidor central e está responsável por recolher automaticamente de *'x'* em *'x'* minutos através do *cron* os binários suspeitos que se encontram no servidor de *mail* para efeitos de identificação e análise dinâmica. Permite ainda enviar os resultados por *e-mail* para os utilizadores que enviaram os binários suspeitos, sob a forma de um alarme simplificado (apenas com os resultados de identificação/detecção e *URL* da análise dinâmica) recorrendo aos subcomponentes *alarm* e *alarmwrite*.

Funcionamento

1. Envio de *e-mail* com o binário suspeito anexado através do *btmclient* para um servidor de *mail*.
2. Recolha automática dos novos binários suspeitos que foram enviados no passo 1 para o servidor de mail, através do componente *mailbinary-robot* que é executado através do *cron* de *'x'* em *'x'* minutos.
3. Para efeitos de identificação/detecção e análise dinâmica do binário suspeito recolhido no passo 2 é efectuada inicialmente uma consulta à base de dados no servidor central para verificar se já existe alguma informação acerca da identificação/detecção e análise dinâmica do respectivo binário. Neste caso evita-se o acesso à plataforma externa *VirusTotal* pois obtém-se a informação necessária directamente da base de dados. Neste caso saltar para o passo 6.
   Se ainda não existir informação na base de dados acerca do binário suspeito em causa, continuar para o passo seguinte.
4. O binário em causa é submetido para a plataforma *VirusTotal* através do sub-componente *avsubmit* (*i.e.* sob controlo do *mwmonitor*) que permite ainda receber os respectivos resultados.
5. O binário em causa é submetido para a plataforma *Anubis* através do sub-componente *ansubmit* (*i.e.* sob controlo do *mwmonitor*) que permite ainda receber os respectivos resultados.
6. Envio automático (por *e-mail*) dos resultados de identificação/detecção por parte dos vários motores de anti-*malware* controlados pela plataforma remota *VirusTotal* e análise dinâmica por parte da plataforma *Anubis*, a partir do servidor central para o servidor de *mail*.
7. Acesso à conta de *mail* para verificação dos resultados de identificação e análise dinâmica do binário suspeito.

- ***artifactsrobot***

  O protótipo *artifactsrobot* também foi escrito em *Python* e permitirá identificar e alertar de forma automática a existência de artefactos de infecção (alterações no *registry* do *Windows*, sistema de ficheiros e objectos de exclusão mútua) em máquinas consideradas críticas.
  No futuro este protótipo será uma mais valia em situações mais avançadas de ataque, por exemplo nas situações em que um *worm* ou um *bot* mais inteligente se propaga pela rede empresarial. Um *worm* ou um *bot* dotado de mais inteligência poderá por exemplo conseguir detectar a presença de

honeypots, evitando assim propagar-se para as sondas de rede onde estes se encontram em execução.

Arquitectura

Através da figura 5.5 pode-se constatar que este protótipo é composto por dois componentes fulcrais, o *artifactsrobot* cuja execução é controlada pelo componente *mwmonitor* no servidor central e o *artifacts* que é executado de *'x'* em *'x'* minutos em cada uma das máquinas consideradas críticas na rede empresarial.

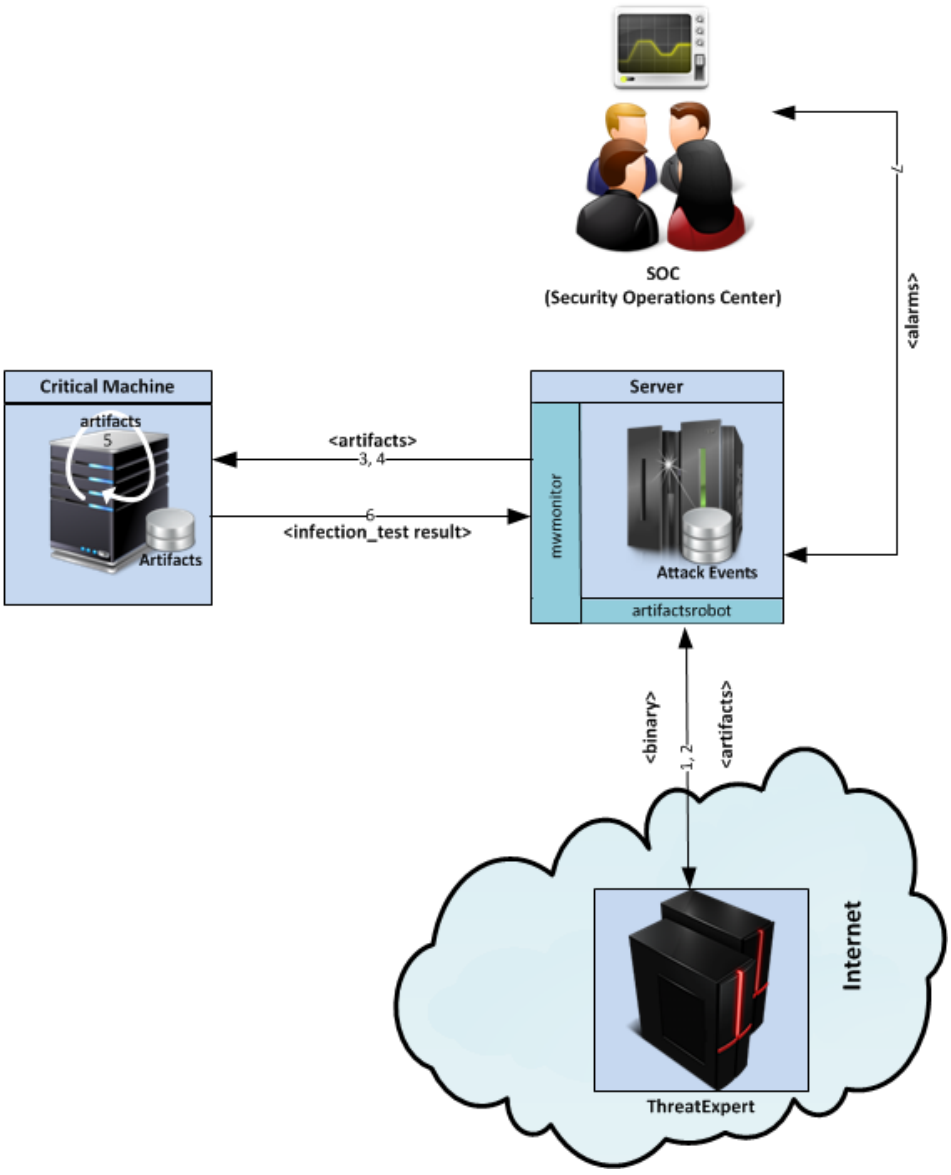


Figura 5.5: *artifactsrobot*

Componentes

- ***artifactsrobot***
  Este componente está responsável por submeter para uma plataforma externa auxiliar (*ThreatExpert*) e recolher os respectivos artefactos ou metadados de análise dinâmica da execução dos binários suspeitos envolvidos em ataques.
- ***artifacts***
  Este componente encontra-se em cada uma das máquinas críticas da rede empresarial. Está responsável por realizar uma pesquisa de *'x'* em *'x'* minutos por vestígios de infecção tendo por base os artefactos ou metadados de análise dinâmica de binários suspeitos obtidos pelo componente *artifactsrobot* no servidor central.

Funcionamento

1. Envio do binário suspeito para a plataforma externa *ThreatExpert* (na execução do *mwmonitor* através do componente *artifactsrobot*).
2. Recolha dos artefactos ou metadados de análise dinâmica da execução do binário suspeito na plataforma externa *ThreatExpert* (através do componente *artifactsrobot*).
3. Envio dos artefactos ou matadados de análise dinâmica de *'x'* em *'x'* minutos para cada uma das máquinas consideradas críticas na rede empresarial.
4. Recolha dos artefactos ou metadados por cada uma das máquinas consideradas críticas e respectivo registo numa base de dados *SQLite*.
5. Pesquisa local de *'x'* em *'x'* minutos em cada uma das máquinas críticas, por vestígios típicos de infecção através de *worms* dotados de mais inteligência.
6. Os resultados da pesquisa são enviados de volta para o servidor central.
7. Alarmística da situação para uma equipa operacional responsável por tratar deste tipo de incidentes (*i.e.* equipa do *SOC*).

5.3.3 Mecanismo de *log*

O componente *dionaea* que executa nas sondas permite armazenar os seus *warnings* e erros de execução em ficheiros de *log* próprios.
O resultado das operações levadas a cabo pelos componentes que constituem o

WMS e pelos protótipos que estendem o *mwmonitor* são escritos no *syslog* do servidor central para monitorização do (in)correcto funcionamento dos componentes do sistema.

5.3.4 *WMSi*

Para um especialista de segurança e/ou administrador de sistemas visualizar e monitorizar os eventos de ocorrência de ataque através da propagação de *malware* em redes empresariais, foi construída uma aplicação *web* (ver figura 5.6) em *PHP* que permite a visualização (através de *dashboards*) de estatísticas e métricas de segurança, através dos dados presentes na base de dados de eventos de ataque do servidor central.

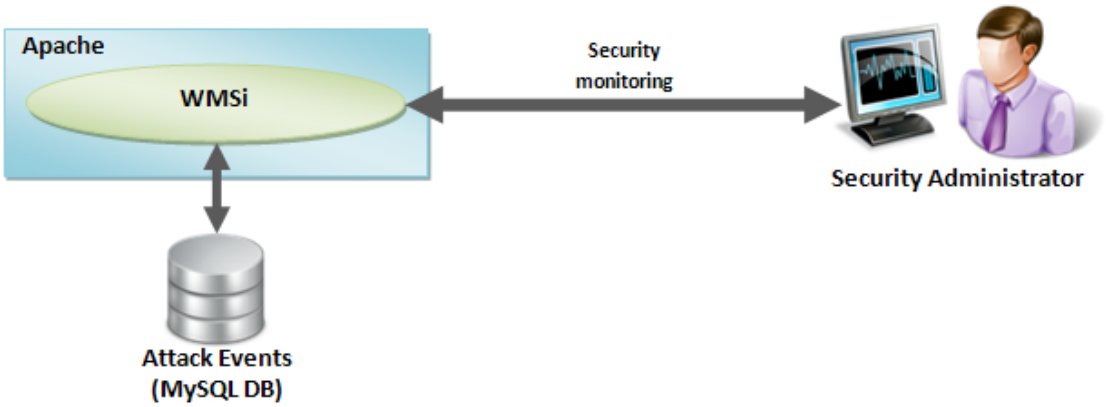


Figura 5.6: *WMSi*

Métricas de segurança

Uma métrica [9] é definida como um valor que pode ser tipicamente um número cardinal, uma média, uma percentagem, ou um valor *booleano*.
As métricas de segurança [14][30] permitem avaliar os níveis de segurança de um sistema, produto ou processo. Sabe-se ainda que a monitorização destina-se em grande medida à construção de métricas para um determinado evento.
Neste caso foram construídas métricas para eventos de ocorrência de ataque originados pela propagação automática de *malware*. Foram idealizadas e implementadas as seguintes métricas de segurança:

- **Número de ligações (suspeitas) por cada fonte de ataque:**
  - Permite monitorizar as fontes de ataque mais frequentes (filtro mensal).

- **Número de ligações por cada porta:**
  - Permite monitorizar as portas mais frequentemente atacadas (filtro mensal).
- **Número de ligações a cada serviço disponibilizado pelo *honeypot*:**
  - Permite monitorizar os serviços mais frequentemente atacadas (filtro mensal).
- **Número de ataques por hora:**
  - Permite monitorizar a distribuição horária da frequência de ataques (filtro diário e mensal).
- **Número de ocorrências de *malware* capturado por cada sonda (*honeypot*) em actividade:**
  - Permite monitorizar as sondas mais activas (filtro diário, mensal e pelos últimos 7 dias de ocorrências) e *malware* mais activo/persistente (filtro por estado *HIGH_PERSISTENCE* e *EXTREME_PERSISTENCE* assim como filtro diário e mensal).

Dashboards

Para visualizar graficamente cada uma das métricas apresentadas anteriormente foram construídos *dashboards* (recorrendo à plataforma *FusionCharts*[9]) que incluem gráficos 2D e 3D de vários tipos (radar, barras verticais, *doughnut* e linha), de acordo com o contexto da métrica em causa.

Attacks Topology Schema

Para visualizar graficamente a topologia dos últimos *'x'* ataques (que foram realizados através da propagação de *malware*) detectados na rede, foi construído um mecanismo que gera dinamicamente um gráfico *dragnode*.
O gráfico *dragnode* é um tipo de gráfico especial com uma interface altamente intuitiva para os utilizadores finais, onde cada conjunto de dados é mostrado como um nó interactivo. Tornando-se perfeito para diagramas de rede, estruturas de hierarquia, etc. Este tipo de gráfico permite ainda uma monitorização rápida e eficiente assim como dá uma ideia geral do panorama da rede no que diz respeito a situações de ataque. Permite ainda realçar através de mudanças visuais (*i.e.* cores, espessura de linhas, etc) o último evento de ataque, assim como o

[9] `http://fusioncharts.com/`

evento de ataque que tiver o estado definido como *HIGH_PERSISTENCE* ou *EXTREME_PERSISTENCE*.
Adicionalmente permite ainda visualizar informação acerca da identificação do binário suspeito envolvido em cada evento de ataque, a informação acerca das máquinas atacantes (*IP*, porta origem, etc), esquematização dos *honeypots* envolvidos na captura dos binários envolvidos nos ataques, o serviço vulnerável explorado no ataque assim como outras informações relevantes, como por exemplo os serviços explorados em cada um desses ataques.

Outros serviços disponibilizados

A aplicação *web* permite ainda mostrar a informação de identificação/detecção do binário suspeito envolvido em cada ataque, assim como o *URL* para os resultados da respectiva análise dinâmica do binário.
Para cada evento de ataque são também mostrados todos os outros dados constituintes (ver tabela 4.1).

5.4 Metodologia

Foi realizada uma análise e selecção de tecnologias que combinadas com os recursos físicos disponíveis se mostraram bastante eficientes no desenvolvimento completo e avaliação experimental do sistema (capítulo 6).

5.4.1 Tecnologias usadas

As tecnologias usadas já foram em grande parte realçadas ao longo do anterior e do presente capítulo, pelo que se segue apenas as tabelas de síntese das tecnologias principais utilizadas pelas plataformas externas auxiliares *VirusTotal* (*VT*) e *Anubis* (ver tabela 5.2), pelos componentes principais do *WMS* (ver tabela 5.3), e por fim pelas extensões ao componente *mwmonitor* (ver tabela 5.4).

	TrID	**PEFile**	**SigCheck**	**PEiD**	**Prevx**	**ThreatExpert**	**Qemu**	**BRO**	**SigBuster**
VT	X	X	X	X	X	X			
Anubis				X			X	X	X

Tabela 5.2: *VT* e *Anubis*: tecnologias usadas.

	C	Python	PHP	VT	Anubis	p0f	cURL	libemu	Apache	MySQL	SQLite	FusionCharts
dionaea	X	X				X	X	X			X	
receivehttp			X						X	X		
mwmonitor		X		X	X					X		
WMSi			X						X	X		X

Tabela 5.3: *WMS*: tecnologias usadas.

	Python	VT	Anubis	ThreatExpert	mwmonitor	MySQL	SQLite
mailrobot	X	X	X		X	X	
artifactsrobot	X			X	X		X

Tabela 5.4: *mailrobot* e *artifactsrobot*: tecnologias usadas.

5.4.2 Recursos disponíveis

Este projecto usufruiu de uma máquina *dual-core* a funcionar como servidor central assim como de duas sondas para capturar a informação pretendida (onde executa em cada uma, o componente que captura tráfego supostamente malicioso, o *dionaea*). É importante realçar o facto do *dionaea honeypot* não exigir a sua execução num sistema computacional de alto custo.

5.5 Resumo

Este projecto tem uma arquitectura distribuída que dispõe de uma sonda (ou na prática, uma federação de sondas) onde executa um *honeypot* para captura de tráfego malicioso e um *handler* que permite o envio desse tráfego para um servidor central.

No servidor central o *receivehttp* recebe a informação que vem da sonda e através do componente *mwmonitor* e dos sub-componentes *avsubmit* e *ansubmit*, obtém automaticamente os dados de identificação e análise dinâmica do binário suspeito capturado no tráfego malicioso (ou tráfego de ataque). De seguida são gerados alarmes (sub-componente *alarm*) para os incidentes de ataque, reportando a situação para uma equipa operacional (sub-componente *alarmwrite*).

Adicionalmente foram criadas métricas de segurança para monitorização do estado de protecção da rede empresarial e extracção de determinados valores que permitem gerar estatísticas ou tendências que a curto prazo poderão e deverão ser uma mais valia na prevenção de incidentes de ataque através da propagação automática de *malware*.

Foi tomada especial atenção à escolha das tecnologias no que diz respeito ao

consumo de recursos (*CPU e RAM*) e tendo em conta os recursos disponíveis, o sistema foi optimizado o melhor possível. Houve um esforço significativo para garantir a modularidade do sistema assim como um *low-coupling* entre os componentes constituintes. Desta forma, será fácil integrar o sistema em produção com outros sistemas. Como prova deste facto, foram idealizados e concretizados dois protótipos de teste para mais duas aplicações (extensões) do sistema, o *mailrobot* e o *artifactsrobot*.

Capítulo 6

Avaliação experimental

Para além do desenvolvimento do modelo teórico apresentado no capítulo anterior, neste capítulo são apresentados detalhes da avaliação experimental levada a cabo para provar a veracidade do modelo teórico perante tráfego intenso através de ataques reais.

Como se sabe, os procedimentos e os danos causados por um ataque real poderão ser imprevisíveis pois a cada dia existem novas formas de ataque ou novas formas de contornar mecanismos de segurança.

Visto tal facto, o modelo teórico do sistema foi implementado e avaliado experimentalmente tendo em conta que poderão existir ataques em que a previsibilidade dos danos provocados numa infra-estrutura de rede sejam nulos ou quase nulos.

6.1 Ambiente e condições

Inicialmente o sistema foi implementado num ambiente empresarial (*i.e.* na rede empresarial da PT), porém, durante o tempo que este esteve sobre testes não houve tráfego suspeito de actividade maliciosa e esta foi uma das limitações para teste e avaliação.

A ideia passou por colocar as sondas ligadas à *Internet*, dando desta forma um ambiente ideal para teste do sistema em produção, só assim se testaram os limites de captura de tráfego suspeito de actividade maliciosa da forma mais real possível ou seja perante uma maior frequência de ataques reais.

O bom funcionamento do sistema num ambiente com um nível de exposição crítico, ou na *Internet*, certamente que dá uma convicção bastante forte de um bom funcionamento em produção numa rede empresarial. Para testar o correcto funcionamento do sistema recorreu-se numa fase inicial à exposição do sistema na rede empresarial ou seja utilizou-se uma sonda com uma instância do *dionaea* e um servidor central com uma instância do componente *receivehttp* e *mwmonitor*,

assim como os seus sub-componentes.

Na rede empresarial, apenas foram realizados ataques simulados (ou injectados através do *Metasploit Framework* [20]) em ambiente controlado.

Após os testes internos e controlados colocou-se o sistema em produção na *Internet* e recorreu-se a duas sondas e a um servidor central para realização da avaliação experimental final ou monitorização a partir das métricas de segurança construídas (capítulo 5).

Foram ainda extraídos vários dados estatísticos e algumas tendências interessantes verificadas na ocorrência de ataques por propagação de *malware* (*i.e.* recorrendo ao *graphic engine* da aplicação web *WMSi*).

6.2 Procedimento

A avaliação experimental teve por base a execução de diversos cenários de uso do *WMS* assim como a execução de outras aplicações do componente *mwmonitor*, como o protótipo *mailrobot* e parcialmente o *artifactsrobot*.

Colocou-se o sistema sobre avaliação na rede empresarial durante o mês de Fevereiro e na *Internet* durante determinados dias dos meses de Março e Abril e depois Junho e Julho, sendo que os resultados mais significativos ocorreram nestes últimos dois meses, daí o facto do relato da avaliação experimental focar-se na execução do sistema apenas para os meses de Junho e Julho, ou seja com as sondas (*honeypots*) a capturarem tráfego e binários suspeitos da *Internet*.

É importante realçar o facto de que todos os gráficos estatísticos que são o resultado das métricas de segurança construídas, foram fielmente copiados para este capítulo a partir da aplicação web *WMSi* implementada especialmente para tais tarefas de monitorização tendo por base o conjunto de dados registados na base de dados *attackevents* do servidor central.

6.3 Cenários de utilização

Para experimentar o sistema foram seleccionados os seguintes cenários de utilização:

1. Captura, identificação/detecção, análise de *malware* propagado e respectiva alarmística (4.1, 4.2, 4.3, 5.3.1, 5.3.2 e 5.3.3).

2. Monitorização e análise de tendências acerca das ocorrências de ataque (5.3.4).

3. Identificação/detecção e análise de ficheiros suspeitos através de *mail attachments* (5.3.2).

4. Verificação baseada em artefactos de infecção em máquinas críticas (5.3.2).

Para cada um destes cenários existe uma fase de preparação e experimentação do sistema (*i.e.* configurações, execução, afinações, etc) e respectiva análise dos resultados (*i.e. execution footprint*, etc).

6.3.1 Cenário 1

No servidor central:

1. Instalou-se o *XAMPP* para *Linux* (*i.e. Apache* com *PHP, MySQL* e *Perl*)[1] e o *Python 2.6.6* numa máquina com o sistema operativo *Debian 5 (Lenny)*.

2. Configurou-se a interface de rede principal (*i.e. ifconfig -s eth0 192.168.2.1 up*).

3. Colocou-se o componente *receivehttp* e *config.ini* em */opt/lampp/htdocs* com as devidas permissões.

4. Configurou-se o componente *receivehttp* em *config.ini*.

   /opt/lampp/htdocs/config.ini

   ```
   1 USER=user
   2 PASSWD=passwd
   3 DB=attackevents
   4 SERVER=localhost
   5 FILEPATH=/usr/share/mwmonitor/binaries/
   ```

   Na linha 1 do *config.ini* é definido o nome de utilizador da base de dados, a palavra chave é definida na linha 2, o nome da base de dados *MySQL* é definido na linha 3, na linha 4 é definido o *hostname* da máquina que alberga a base de dados, neste caso a base de dados encontra-se instalada localmente (*localhost*), e por fim é definido na linha 5, o caminho para o repositório local de binários.

5. Colocou-se o componente *mwmonitor* e os seus sub-componentes em */usr/share/mwmonitor/core*.

6. Configurou-se o componente *mwmonitor* e sub-componentes *mysql* e *ansubmit* em *config.ini*.

   /usr/share/mwmonitor/core/config.ini

   ```
   1 [monitor]
   2 filespath=../binaries/
   3
   ```

[1] http://apachefriends.org/en/xampp-linux.html

```
4  [mysql]
5  host=localhost
6  port=3306
7  user=user
8  passwd=passwd
9  db=attackevents
10 unix_socket=/opt/lampp/var/mysql/mysql.sock
11
12 [anubis]
13 email=email
14 user=user
15 passwd=passwd
16 analysis_type=FILE
```

Nas linhas 1 e 2 é configurado o *mwmonitor* no que diz respeito ao caminho relativo para o repositório dos binários recebidos. Seguidamente nas linhas 4-10 são definidos os parâmetros de configuração acedidos pelo sub-componente *mysql* para ligação à base de dados local (*attackevents*).
Por fim nas linhas 12-16 são definidos os parâmetros de configuração do sub-componente *ansubmit* para análise dinâmica dos binários localizados em */usr/share/mwmonitor/binaries* assim como o respectivo envio do *URL* com os resultados para o endereço de *e-mail* definido na linha 13.

7. Configurou-se o sub-componente *avsubmit* editando a variável *VTAPIKEY* (no código) com uma *private key* que permite o acesso sem restrições ao *VirusTotal*.

8. Configurou-se o *cron service* para execução automática do componente *mwmonitor*.

crontab

```
1 */5 * * * * bash -c "cd /usr/share/mwmonitor/core/;/usr/share/
    mwmonitor/core/mwmonitor.py"
```

Na linha 1 encontra-se a instrução (introduzida através de *crontab -e*) que permite executar de *'x'* em *'x'* minutos (neste caso de 5 em 5 minutos) o componente *mwmonitor*.

Na(s) sonda(s):

1. Instalação do *dionaea honeypot*.

Existem inúmeros pacotes que devem ser rigorosamente instalados para que o dionaea funcione correctamente.
Muitos destes pacotes requerem a compilação a partir dos respectivos *sources*

pois grande maioria das dependências não está disponível a partir do *package manager* local. Os passos de instalação não são apresentados neste documento, pois todo o processo está muito bem documentado na página *web* do projecto *dionaea*[2]. Depois de instalado com sucesso o *dionaea*, os respectivos ficheiros devem estar presentes em */opt/dionaea*. Este *path* a partir daqui será referido por *$DIONAEA_HOME*.

2. Configuração do *dionaea honeypot*.

   Antes da execução do *dionaea*, este deve ser configurado em: *$DIONAEA_HOME/etc/dionaea/dionaea.conf*.

   Neste ficheiro existem várias secções de configuração que já foram resumidamente apresentadas no capítulo anterior, estas são apresentadas de seguida em maior detalhe.

   (a) A secção *Logging*

   O *dionaea* escreve por omissão mensagens para *debug*, mensagens de informação, *warning* e de erro em ficheiros de *log* próprios. Este facto é importante para efeitos de teste mas pode ser bastante pesado em termos computacionais para efeitos de produção, nesta situação iríamos ter ficheiros de *log* muito grandes em curto prazo.
   Esta foi uma das preocupações aquando da configuração do *honeypot* na(s) sonda(s), mesmo que se trate aqui apenas de uma actividade experimental. Visto isto, foram realizadas as seguintes alterações na secção de *Logging* (*Log Level*) do ficheiro de configurações do *dionaea*:

Parâmetros *"default"*	
Valor original:	**Novo valor:**
levels = "all"	*levels = "all, -debug"*
Parâmetros *"errors"*	
Valor original:	**Novo valor:**
levels = "warning, error"	*levels = "error"*

Tabela 6.1: Alterações no *Log Level* do *dionaea*.

Como se pode constatar na tabela 6.1 evita-se assim a escrita de mensagens de *debug* e *warning* para ficheiros de *log*, melhorando significativamente o desempenho do *dionaea* na sonda.

[2] `http://dionaea.carnivore.it/#compiling`

(b) A secção *IP*

Como já foi referido o *dionaea*, por omissão, faz *bind* a todos os endereços *IP* configurados usando ambos *IPv4* e *IPv6*. Alterou-se a configuração do parâmetro *IP bind mode* que estava definido por omissão como *getifaddrs* (*bind* a todos os endereços *IP* configurados) para o modo *manual* (*mode = manual*), em que temos a liberdade de definir as interfaces de rede e os endereços *IP* que desejarmos. Seguem-se as formas de configuração manual mais utilizadas (os endereços *IP* seguintes servem apenas para demonstração):

```
1  mode = "manual" //antes "getifaddrs"
2  addrs = {eth0 = ["0.0.0.0"]}
3  addrs = {eth0 = ["10.14.49.50", "10.14.49.51"]}
4  addrs = {eth0 = ["10.14.49.50"], eth1 = ["0.0.0.0"]}
5  addrs = {eth0 = ["::"]}
6  addrs = {eth0 = ["::"], eth0 = ["0.0.0.0"]}
```

Dependendo do número de interfaces de rede e endereços *IP* configurados, o modo *getifaddrs* pode ser rápido, quando existem poucas interfaces de rede, caso contrário, o processo de *binding* pode tornar-se um pouco lento.
Na linha 1 o *dionaea* faz *bind* a todos os endereços *IPv4* na interface *eth0*, na linha 2 faz-se *bind* dos endereços 10.14.49.50 e 10.14.49.51 na interface *eth0*, na linha 3 é feito *bind* ao endereço 10.14.49.50 na interface *eth0* e a todos os endereços *IPv4* na interface *eth1*, a linha 4 é equivalente à linha 1 mas para *IPv6*, e por fim na linha 5 é feito *bind* a todos os endereços *IPv4* e a todos os endereços *IPv6* na interface *eth0*. Neste projecto para efeitos de teste foram utilizadas todas as formas, no entanto a mais utilizada foi a da linha 2, mas numa versão apenas com um endereço *IP* para cada sonda da federação. Como exemplo real, no dia 26-6-2011 pelas 05:52:24 o *dionaea* tinha o endereço 95.69.29.39 associado a uma interface *ppp0* numa das sondas:

```
1  mode = "manual"
2  addrs = {ppp0 = ["95.69.29.39"]}
```

(c) A secção *Module*

Nesta secção pode-se activar, desactivar e configurar várias funcionalidades e ferramentas usadas pelo *dionaea honeypot*. Como já foi referido neste documento existem duas sub-secções mais relevantes, a *ihandlers*

e a *services*.
Em baixo pode-se ver a configuração das sub-secções *ihandlers* (linhas 1-12) e *services* (linhas 14-22) mais frequente na(s) sonda(s) que estiveram sob actividade experimental neste projecto.

```
1  ihandlers = {
2      handlers = ["ftpdownload",
3                  "tftpdownload",
4                  "emuprofile",
5                  "cmdshell",
6                  "store",
7  //              "uniquedownload",
8                  "logsql",
9  //              "logxmpp",
10                 "p0f",
11 //              "surfids"]
12 }
13
14 services = {
15     serve = ["http",
16              "https",
17              "tftp",
18 //           "ftp",
19              "mirror",
20              "smb",
21              "epmap"]
22 }
```

Como o *dionaea* usa uma pequena base de dados *SQLite* local, o *logsql handler* (linha 8) está activo por omissão. Depois existem outros *handlers* importantes para *download* dos binários via *FTP* (linha 2) e *TFTP* (linha 3), na linha 4 encontra-se activo o *emuprofile handler* que recorre à *libemu* para detecção de *shellcode*. Também está activo o *cmdshell handler* (linha 5) que fornece uma *shell* muito simples ao atacante. Depois na linha 6 encontra-se activo um *handler* (*store handler*) que permite a captura e salvaguarda dos binários para depois serem submetidos para o servidor central (configuração na secção *Submit*). O facto de se optar por uma arquitectura descentralizada com uma federação de sondas a comunicar com o servidor central, levou a que se desactivasse o *handler* que permite apenas que o *dionaea* capture o mesmo binário (*MD5*) uma única vez, mesmo que venha de fontes de ataque distintas (*SourceHost*). Como esse controlo passou a ser feito no servidor central pelos componentes *receivehttp* e *mwmonitor* (relembrando que um ataque único é definido pelo tuplo *SourceHost, TargetPort e MD5* do binário) não faz sentido o *uniquedownload handler* estar activo (linha 7). Por fim, integraram-se os serviços do *p0f* com o *dionaea honeypot*, daí o

p0f handler estar activo (linha 10). Na sub-secção *services* quase todos os serviços foram activados à excepção do serviço *ftp*.
O *smb* e o *epmap* são essenciais na captura de binários suspeitos de actividade maliciosa com o *dionaea*, vale a pena realçar que grande parte do *malware* ataca frequentemente os serviços *smb* e *epmap*. O serviço *tftp* funciona como um servidor que aceita transferências de ficheiros arbitrários e também detecta tentativas de exploração de vulnerabilidades contra o mesmo serviço.
Os serviços *http* e *https* agem como um servidor *web* que serve os ficheiros que se encontram em *$DIONAEA_HOME/var/dionaea/wwwroot/*.
Por fim o serviço *ftp* permite o *login* por parte do atacante assim como a captura de ficheiros que este possa transferir. Este serviço foi desactivado pela simples razão de ainda não detectar *exploits* para si dirigidos e na prática transformar a sonda num servidor de ficheiros para a *Internet* pode ser bastante perigoso.

(d) A secção *Submit* (*submithttp*)

```
1  submit_http = {
2      url = "http://192.168.2.1/receivehttp.php"
3      email = "email"
4      user = "lampp"
5      pass = "passwd"
6      file_fieldname = "upfile"
7      MAX_FILE_SIZE = "1500000"
8      submit = "Submit for analysis"
9  }
```

Na linha 2 define-se o *URL* do servidor central para efeitos de recepção e armazenamento das *fingerprints* de ataque e respectivos binários suspeitos de actividade maliciosa. De seguida é configurado o endereço de *e-mail* que receberá as notificações de erro. Caso o *Apache* (servidor *HTTP*) no servidor central necessite de autenticação, então deve ser configurado na linha 4 o nome de utilizador e na linha 5 a palavra chave associada. Assim como na linha 2, a configuração dos parâmetros *file_fieldname* e *MAX_FILE_SIZE* (linhas 6 e 7) são obrigatórios. Por fim na linha 8 é configurada a descrição associada à respectiva operação, neste caso, uma operação de submissão de conteúdo.

(e) *Log Rotation*

/etc/logrotate.d/dionaea

```
1  /opt/dionaea/var/log/dionaea*.log {
2      notifempty
```

```
3         missingok
4         rotate 28
5         daily
6         delaycompress
7         compress
8         create 660 root root
9         dateext
10        postrotate
11            kill -HUP `cat /opt/dionaea/var/run/dionaea.pid`
12        endscript
13    }
```

3. Instalação, configuração e execução do *p0f* para integração com o *dionaea honeypot*.

   Como já foi referido o *dionaea honeypot* suporta a integração do *p0f* para que se identifique passivamente o sistema operativo da fonte de ataque. Depois da instalação do *p0f* (*i.e. sudo apt-get install p0f*). Os resultados obtidos serão registados na base de dados *SQLite* e por omissão o *dionaea* é configurado para ler os dados coleccionados pelo *p0f* através de *Unix domain socket* (para comunicação entre processos) criado em */tmp/p0f.sock*. Para iniciar o *p0f* como um *daemon* de maneira a que este seja usado pelo *dionaea*, executou-se o seguinte comando como *root*:

```
1  $ sudo p0f -i any -u root -Q /tmp/p0f.sock -q -l \
2    -d -o /dev/null -c 1024
```

   Os parâmetros usados na execução do *p0f* são referidos na tabela 6.2.

Parâmetro	Descrição
-i *any*	as interfaces que ficarão à escuta (*any* para todas as interfaces disponíveis)
-u *root*	*chroot* e *setuid* para *root*
-Q */tmp/p0f.sock*	cria um *Unix domain socket* com um nome específico
-q	não mostrar o *banner*
-l	usar uma única linha de *output*
-d	executar o *p0f* como um *daemon*
-o */dev/null*	enviar todo o output para */dev/null*
-c *1024*	tamanho da *cache* usada em -Q

Tabela 6.2: *p0f*: parâmetros de execução.

Como o *dionaea honeypot* será executado como um *daemon* com a conta *nobody*, foi realizada a seguinte alteração no ficheiro *p0f.sock*:

```
$ sudo chown nobody:nogroup /tmp/p0f.sock
```

4. Execução do *dionaea honeypot*.

   Para iniciar o *dionaea honeypot* como um *daemon* executou-se o seguinte comando como *root*:

```
$ sudo ./dionaea -u nobody -g nogroup \
  -p /opt/dionaea/var/dionaea.pid -D

Dionaea Version 0.1.0
Compiled on Linux/x86 at Dec 26 2010 22:28:54 with gcc 4.4.5
Started on debian running Linux/i686 release 2.6.35-22-generic

[16092011 23:50:15] dionaea dionaea.c:245: User nobody has uid
    65534

[16092011 23:50:15] dionaea dionaea.c:264: Group nogroup has gid
    65534
```

   A descrição dos parâmetros de execução está muito bem documentada para consulta na página *web* do projecto *dionaea*[3].
   Verificou-se ainda, após execução do comando *ps aux*, que o *dionaea honeypot* consumiu na sua execução como um *daemon* uma média de apenas 0.3% de *CPU* e 2.8% de *RAM*.

5. Alguns dados obtidos pelo *p0f*.

```
$ sudo /opt/dionaea/var/dionaea/logsql.sqlite
sqlite> select p0f, p0f_genre, p0f_link, p0f_detail from p0fs
    limit 5;
1|Windows|ethernet/modem|2000 SP4, XP, SP1+
2|Windows|pppoe (DSL)|XP/2000 (RFC1323+, w+, tstamp+)
3|Windows|ethernet/modem|2000 SP4, XP, SP1+
4|Windows|IPv6/IPIP|2000 SP4, XP SP1+
5|Windows|pppoe (DSL)|XP/2000 (RFC1323+, w+, tstamp+)
```

[3] http://dionaea.carnivore.it/#running

No servidor central:

Após captura de tráfego suspeito de actividade maliciosa com o *dionaea honeypot* nas sondas 95.69.29.39 em 26 de Junho e 188.140.2.132 em 8 de Julho, este é enviado para o servidor central que controla os processos de identificação e análise dinâmica do binário envolvido no tráfego recebido e por fim como exemplo real produz os seguintes alarmes de ataque que corresponderam ao eventos ocorridos:

/usr/share/mwmonitor/core/mwalarm.log

```
1 [192.168.2.2, 26fe6e1c61e63e0a9fa52c4f24b7a67e, 212.233.155.26, 4298,
    95.69.29.39, 445, microsoft-ds, 2011-06-26 05:40:40, http
    ://212.233.155.26:1824/cfke, Net-Worm.Win32.Kido.ih, W32.Downadup.B
    , Artemis!26FE6E1C61E6, a variant of Win32/Conficker.X, W32/
    Conficker.C.worm, Worm:Win32/Conficker.B, Win32:Rootkit-gen, Worm/
    Downadup, Trojan.Dropper-18535, http://anubis.iseclab.org/?action=
    result&task_id=119682769d321aa84d58d55b914a568b4]
```

```
1 [192.168.2.6, ebbfa21230fdda9eccf16ae4295534d3, 188.140.130.12, 53499,
    188.140.2.132, 445, microsoft-ds, 2011-07-08 18:08:59, ftp://ccc:1
    @60.10.179.100:5809/bb6.jpg, Net-Worm.Win32.Kolab.acem, W32.IRCBot.
    Gen, W32/Spybot.worm!ed, Win32/AutoRun.IRCBot.HY, Generic Malware,
    Backdoor:Win32/Sdbot, Win32:Malware-gen, Worm/Generic2.AQNH, PUA.
    Packed.NPack-2, http://anubis.iseclab.org/?action=result&task_id
    =143b2ba9e777746849d330ff7964a1fa2]
```

Estes alarmes (ver formato nas tabelas 4.1 e 5.1) foram gerados e guardados localmente no servidor central, no entanto facilmente poderão ser submetidos para outra máquina e/ou aplicação remota.
Como já foi referido no capítulo anterior, o estado de execução das operações do componente *mwmonitor* é sempre escrito em *syslog*. Neste caso, aquando da execução do componente *mwmonitor* na identificação, análise dinâmica e produção de um alarme para a ocorrência de um ataque que envolveu um binário malicioso cujo *MD5* é *c26c352987bc9f678957c89c046a0cdc*, foi utilizado o comando *tail -f /var/log/syslog*, o resultado de *log* pode ser visto em baixo.

/var/log/syslog

```
1 Sep 20 19:35:44 probe mwmonitor: 'Identification/analysis to file
    c26c352987bc9f678957c89c046a0cdc'
2 Sep 20 19:35:44 probe mwmonitor: 'Finished processing'
3 Sep 20 19:35:44 probe mwmonitor: '(192.168.2.2,188.175.77.44,135,
    c26c352987bc9f678957c89c046a0cdc,2011-03-17 11:44:26)
    ALARM_SEND_SUCCESS - State = ALARM'
```

Depois de alguns testes de execução do sistema neste cenário, constatou-se em primeiro lugar, através da integração do *p0f* no *dionaea*, a frequência de ataques

vindos de máquinas com *Windows 2000 e XP*, o que prova desde já um grau de vulnerabilidades de segurança bastante elevado nestas versões do *Windows*.
Outra curiosidade interessante que foi verificada teve a ver com o facto de muitas vezes se ter que aguardar pelo menos cinco minutos por uma ligação suspeita ao *dionaea honeypot*, no entanto a partir da primeira ligação, muitas outras e não directamente relacionadas foram realizadas com maior frequência e com um intervalo temporal bastante curto entre elas.
O período inicial de espera (pelo menos cinco minutos) até à primeira ligação suspeita nos testes experimentais (ligação vinda da *Internet*) pode dever-se ao facto do endereço *IP* do *honeypot* que executa na(s) sonda(s) não ser conhecido à priori pelos atacantes.
Foram ainda medidos vários tempos (variando o número de binários) do processo completo (identificação do binário com o *VirusTotal* através do *avsubmit*, análise dinâmica com o *Anubis* através do *ansubmit* e alarmística do evento de ataque através do *alarm* e do *alarmwrite*) por parte do componente *mwmonitor* no servidor central (ver tabela 6.3).
Para a medição dos tempos de execução foi utilizada a aplicação *time* dos sistemas *Unix* (*time ./mwmonitor.py*). No *output* obtido apenas foi seleccionado o *real execution time* que consiste neste caso, no tempo total utilizado pelo componente *mwmonitor*, desde a sua execução até à sua finalização. De notar que estes tempos foram medidos tendo por base uma ligação à *Internet* com velocidades de *download* até *3,6 MB/s* e *upload* até *384 KB/s*. Para 1, 5 e 10 binários houve sempre o respectivo envio para as plataformas externas, para 229 binários, como logicamente muitos deles vão sendo identificados e analisados ao longo da execução do *mwmonitor*, quando existirem binários repetidos não existe necessidade de envia-los de novo para as plataformas externas na *Internet*, mas sim consultar apenas a base de dados local (*attackevents*) para obter os respectivos resultados de identificação e análise dinâmica.

#Binários	***Real execution time***
1	6.951s
5	30.899s
10	1m14.656s
229	4m34.461s

Tabela 6.3: Medição do *real execution time* para o componente *mwmonitor*

6.3.2 Cenário 2

Tendo em conta a preparação e experimentação do cenário anterior, colocou-se a aplicação *web WMSi* (ver figura 6.1) em */opt/lampp/htdocs* no servidor central. Toda a experimentação e posterior análise dos resultados deste cenário teve por base a aplicação *web WMSi* (concebida especialmente para este projecto) que permite a monitorização das ocorrências de *malware* propagado automaticamente (*worms* e *bots*). Foi analisada a informação obtida nos meses de Junho e Julho a partir das métricas de segurança construídas, e posteriormente foi feita uma extracção de conhecimento a partir dos vários tipos de gráficos produzidos automaticamente a partir dos dados presentes na base de dados *attackevents* do servidor central. Para cada mês e para efeitos de relato da experiência, são apresentadas apenas as sondas que capturaram um maior volume de tráfego suspeito de actividade maliciosa e que disponham simultaneamente de dados interessantes para análise. Visto isto, podemos ver as sondas mais relevantes nos meses de Junho e Julho e em determinados dias destes meses, assim como o número de ligações suspeitas de actividade maliciosa recebidas. Depois segue-se uma análise tendo em conta as várias métricas e os vários tipos de gráficos produzidos para os meses de Junho e Julho. É ainda apresentado o gráfico dos últimos 7 dias em que ocorreram ataques (6.14) assim como um gráfico com o número de ocorrências de ataque nos vários meses do ano em que o sistema esteve em actividade experimental (ver figura 6.15). Por fim são apresentados os resumos dos resultados de análise dinâmica realizada pela plataforma *Anubis* para os binários maliciosos mais activos neste cenário. Na descrição deste cenário é importante ainda realçar que as nomenclaturas dos binários maliciosos propagados nos ataques foram obtidas automaticamente a partir de vários motores de anti-*malware* (no processo de identificação/detecção) mas apenas foram seleccionadas para a escrita deste capítulo, as nomenclaturas atribuídas pelo *Kaspersky* anti-*malware* e pelo *Symantec Norton* anti-*malware*.

Monitorização no mês de Junho

No mês de Junho obtiveram-se alguns resultados interessantes mas não tão variados como os de Julho que serão vistos mais à frente. Os resultados foram interessantes pelo simples facto de haver apenas uma fonte de ataque 201.43.60.111 que se destacou muito mais que as outras em termos de ligações para o *dionaea honeypot*.
Esta fonte de ataque proveniente do Brasil, efectuou 42 dos 64 ataques obtidos (ver figura 6.15), faz parte, como seria de prever, de um evento de ataque classificado como *EXTREME_PERSISTENCE*.

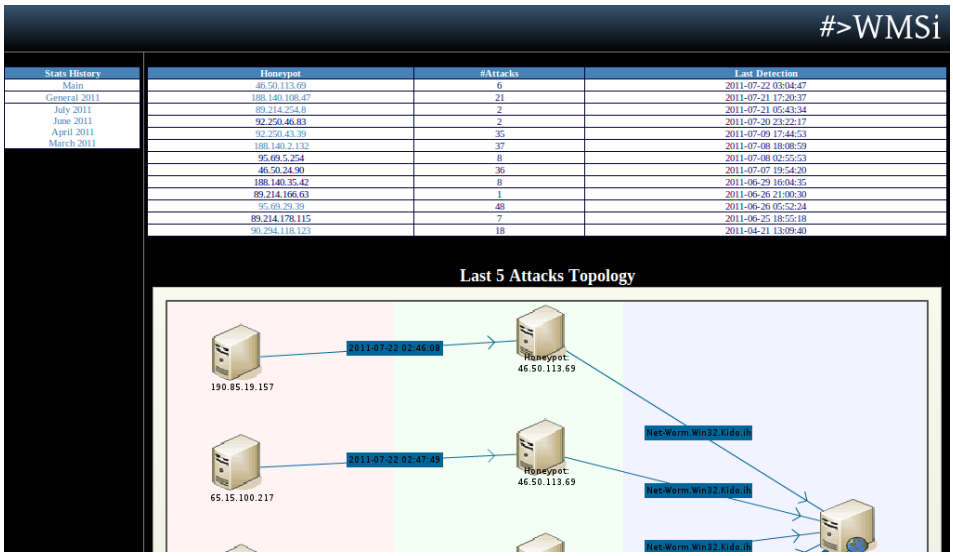


Figura 6.1: *WMSi* - página de entrada

Os ataques exploraram o serviço vulnerável *microsoft-ds* (ver figura 6.5) na porta 445 (tipicamente presentes em máquinas com o sistema operativo *Windows XP unpatched*), incluindo sempre o binário malicioso *Net-Worm.Win32.Kido.ih*.
Como é fácil constatar através da figura 6.4 o *Net-Worm.Win32.Kido.ih* ou *Conficker* (capítulo 3), cujo *MD5* é *a95ca1b2083f0acdc015a7589d5dadda*, abundou nos eventos de ataque (ver figura 6.11 com o sumário da respectiva análise dinâmica), constituindo 71.88% do total de binários maliciosos capturados nos ataques.
Outros binários maliciosos também surgiram nos testes experimentais do mês de Junho, como por exemplo o *Net-Worm.Win32.Kolab.acem* que deteve 17.19% de popularidade, o *Backdoor.Win32.Rbot.bqj* com 4.69%, o *Trojan.Win32.Genome.slaz* e o *Backdoor.Win32.Ruskill.lk* com 3.13%. Através da figura 6.3 podemos afirmar que à excepção da máquina atacante 201.43.60.111 todas as outras (89.214.88.172, 188.140.11.243, etc) realizaram 2 ataques no máximo, com a grande maioria a realizar apenas um único ataque aos serviços *msrpc* (porta 135) e *unix-status* (porta 1957) como se pode ver pela figura 6.5. É importante realçar um facto curioso, pois entre as 05:16:23 e as 05:52:24 horas recebeu-se mais ataques do que nas restantes horas do dia, contribuindo este pequeno intervalo horário para 48 dos 64 ataques obtidos.

Monitorização no mês de Julho

O mês de Julho foi bastante rico em termos de variedade de ataques e de *malware* capturado. Como se pode constatar na figura 6.15 o mês de Julho foi o mês em que existiram mais ocorrências de ataque (147 ocorrências), vale a pena realçar

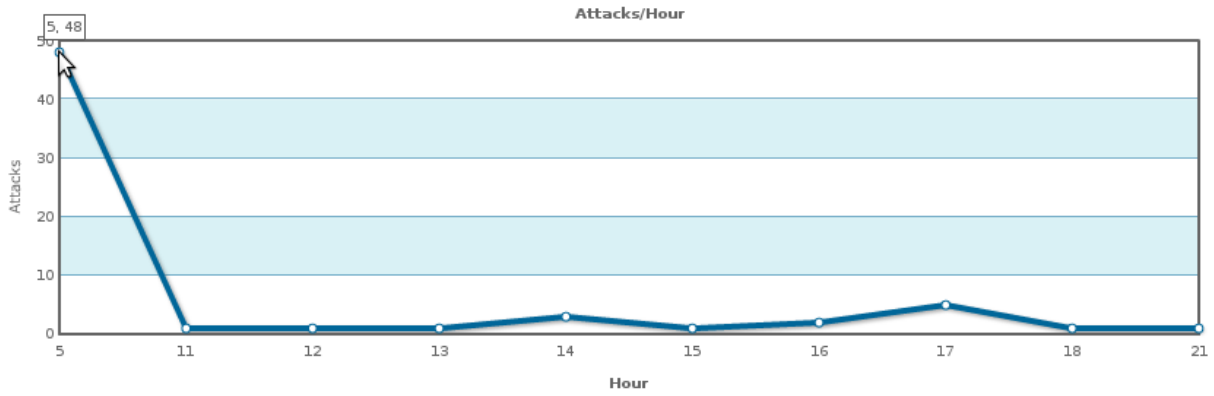


Figura 6.2: Ocorrências de ataque por hora no mês de Junho

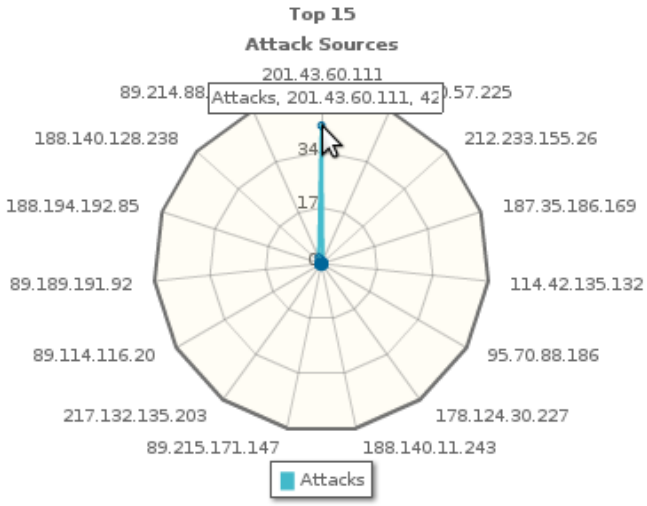


Figura 6.3: Fontes de ataque no mês de Junho

que o sistema não esteve em execução meses completos mas apenas alguns dias e não necessariamente consecutivos. Pela figura 6.6 verifica-se um menor volume de ocorrências de ataque durante as 03:00:00 e as 03:59:59 horas e durante as 06:00:00 e as 07:59:59 horas, e um maior volume entre as 18:00:00 e as 18:59:59 horas. No entanto ao longo dos testes experimentais verificou-se quase sempre picos de elevado volume de ocorrências de ataque e consequente captura de *malware* propagado entre as 11:00:00 e as 11:59:59 horas. Neste caso pode-se verificar que existiu um pico que passou de 6 ocorrências de ataque entre as 10:00:00 e as 10:59:59 horas para 14 ocorrências de ataque entre as 11:00:00 e as 11:59:59. Entre as 15:00:00 e as 15:59:59 horas obteve-se mais duas ocorrências de ataque em relação às 11:00:00-11:59:59 horas e mais duas ocorrências de ataque entre as 18:00:00 e as 18:59:59 em relação às 15:00:00-15:59:59 horas. Este padrão verificado no mês de Julho deve-se à forte actividade do *Net-Worm.Win32.Kolab.acem* ou *W32.IRCBot.Gen* (ver figura 6.12 com sumário da respectiva análise dinâmica),

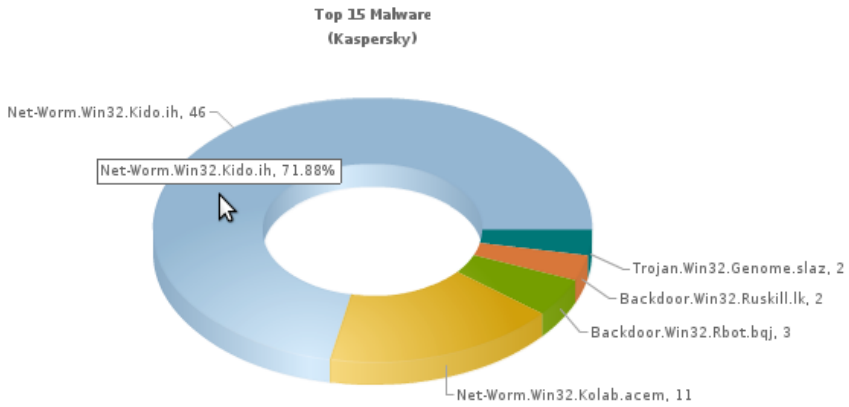


Figura 6.4: *Malware* identificado/detectado e analisado no mês de Junho

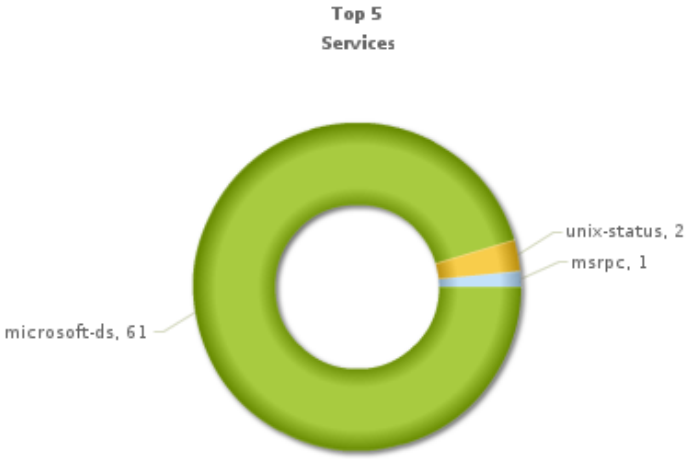


Figura 6.5: Serviços atacados no mês de Junho

que, como se pode constatar na figura 6.8, 61 das 147 ocorrências de ataque obtidas (que equivale a 42.36% do total de ocorrências) consistiram em ataques que recorreram a este *bot* (capítulo 3).
Capturou-se 19 (13.19%) variantes do *Kolab*, neste caso o *Net-Worm.Win32.Kolab.anen*, 11 (7.64%) ocorrências do *Trojan.Win32.Midgare.ayfl* (ver figura 6.13 com sumário da respectiva análise dinâmica), este foi classificado como um *trojan horse* mas na realidade trata-se de um *worm* que curiosamente não foi identificado pelo *Symantec Norton anti-malware* nem pelo *ClamAV*, provavelmente derivado à utilização interna de mecanismos de evasão, por exemplo, recorrendo a *packers* que alteram a estrutura interna do binário malicioso.
Obtiveram-se ainda, mas em menor volume (2 ocorrências ou 1.39% do total de *malware* capturado) outras variantes do *Conficker worm* (capítulo 3), como por exemplo o *Net-Worm.Win32.Kido.dam.am*. O maior volume de ligações maliciosas

teve origem na máquina 188.140.33.102 como se pode verificar na figura 6.7, de seguida houve um evento de ataque classificado como *EXTREME_PERSISTENCE* com origem de ligação na máquina 176.14.168.247.
As portas que receberam mais ligações para ataque foram a 445, 1957 e 135 onde são disponibilizados serviços como *microsoft-ds*, *unix-status* e *msrpc* respectivamente (ver figura 6.9).
Note-se neste caso a supremacia do *Net-Worm.Win32.Kolab.acem* (que foi obtido a partir de *ftp://ccc:1@60.10.179.100:5809/bb6.jpg*) sobre o *Net-Worm.Win32.Kido.ih* (uma variante do *Conficker worm* bastante frequente) que detém 15.28% do total de *malware* capturado. O facto do serviço *microsoft-ds* ser tão explorado só prova o perigo constante que circula na *Internet* quando se dispõe de máquinas com o sistema operativo *Windows 2000 ou XP* mesmo com o *SP2* instalado.
Por fim na figura 6.10 estão esquematizados os últimos 5 ataques que ocorreram no mês de Julho (*Attack Topology Schema*).
As ligações a cinzento indicam que se trata de um evento de ataque para o qual ainda não foi gerado o respectivo alarme, as ligações a azul indicam que já foi produzido um alarme para um evento de ataque ocorrido, as ligações a laranja indicam um evento de ataque classificado com estado *HIGH_PERSISTENCE* e a vermelho um evento de ataque classificado como *EXTREME_PERSISTENCE*, a ligação desenhada com maior espessura indica o último evento de ataque ocorrido.

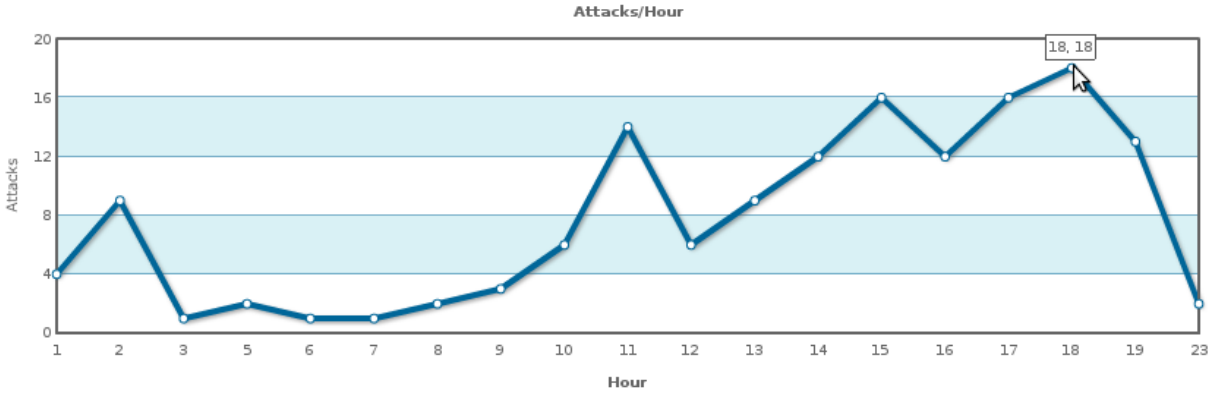


Figura 6.6: Ocorrências de ataque por hora no mês de Julho

A sonda 95.69.29.39 em actividade no dia 26 de Junho de 2011

A sonda 95.69.29.39 esteve em actividade no dia 26 de Junho de 2011 entre as 05:15:23 e as 05:53:00 horas e obteve 48 ataques dirigidos a si através de propagação de *malware*.
Foi obtido um evento de ataque às 05:23:36 horas classificado posteriormente no

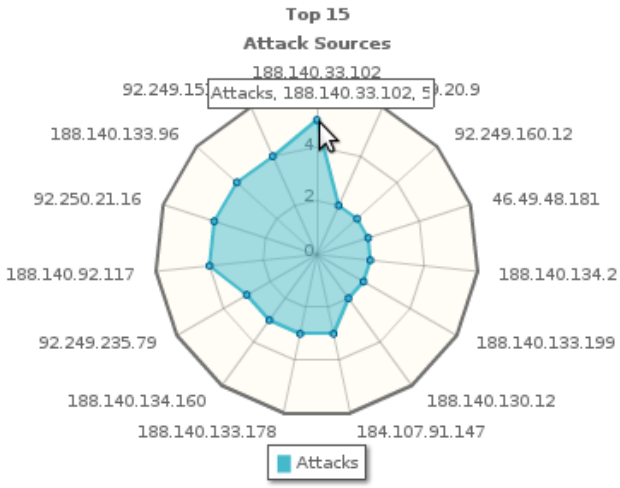


Figura 6.7: Fontes de ataque no mês de Julho

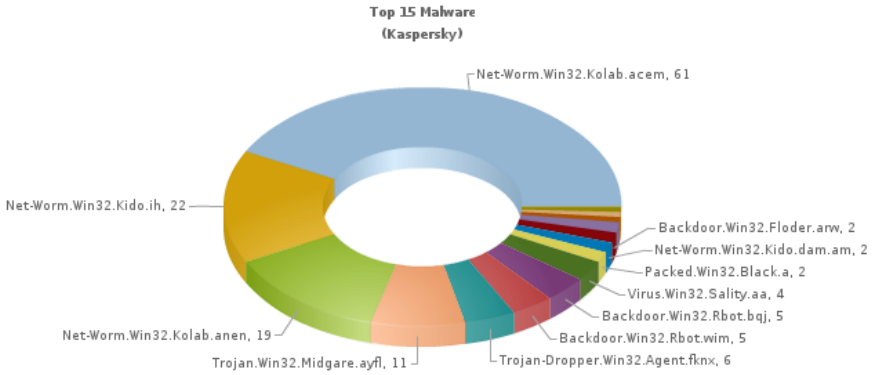


Figura 6.8: *Malware* identificado/detectado e analisado no mês de Julho

servidor central com estado *EXTREME_PERSISTENCE*, teve origem na máquina 201.43.60.111 com sistema operativo *Microsoft Windows XP* (segundo os resultados obtidos pelo *p0f*), atacou o serviço *microsoft-ds* do *Microsoft Windows* (na porta 445) através da vulnerabilidade *MS08-067* com o *Net-Worm.Win32.Kido.ih* ou *Conficker* (capítulo 3) obtido pela sonda a partir do *URL http://201.43.60.111:3209/vcoy*. Uma nota curiosa acerca da persistencia deste ataque tem a ver com o facto de, oito minutos e treze segundos após iniciar o *honeypot* o evento ter sido classificado como *EXTREME_PERSISTENCE*, ou seja já existiam 9 eventos de ataque iguais com origem na mesma máquina atacante (201.43.60.111). Após trinta e seis minutos e um segundo já existiam 42 eventos de ataque iguais, com origem na máquina 201.43.60.111, ou seja apenas 3 eventos de ataque não tiveram origem nesta máquina que propagou o *Net-Worm.Win32.Kido.ih*.
É importante ainda realçar o facto de haver um reduzido intervalo de tempo entre ataques consecutivos e similares, em média, cerca de um minuto (possível tenta-

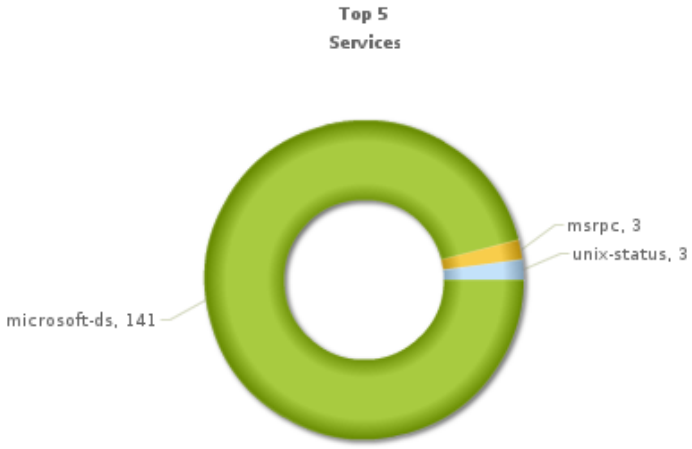


Figura 6.9: Serviços atacados no mês de Julho

tiva de ataque de negação de serviços).

A sonda 188.140.2.132 em actividade no dia 8 de Julho de 2011

A sonda 188.140.2.132 esteve em actividade no dia 8 de Julho de 2011 entre as 09:45:22 e as 18:13:05 horas e obteve 37 ataques dirigidos a si através de propagação de *malware*.
O evento de ataque que mais se destacou teve origem na máquina 188.140.33.102 às 10:18:09 horas e posteriormente ocorreram 5 ataques iguais com origem na mesma máquina, sendo o evento de ataque classificado às 10:27:47 horas como *HIGH_PERSISTENCE*.
O binário malicioso envolvido nos ataques foi o *Net-Worm.Win32.Kolab.acem* (ver capítulo 3), obtido a partir de *ftp://ccc:1@60.10.179.100:5809/bb6.jpg*.
Verificou-se ainda que a máquina atacante tinha instalado o sistema operativo *Windows 2000*, muito provavelmente seria uma máquina *bot/zombie* que faria parte de uma *botnet*. Na sonda 188.140.2.132 verificou-se ainda que o maior volume de tráfego capturado ocorreu entre as 11:00:00 e as 12:00:00 horas.

6.3.3 Cenário 3

Este cenário está relacionado com o cenário 1, sendo assim são omitidos aqui alguns passos e respectivos detalhes de preparação que já foram apresentados no primeiro cenário.

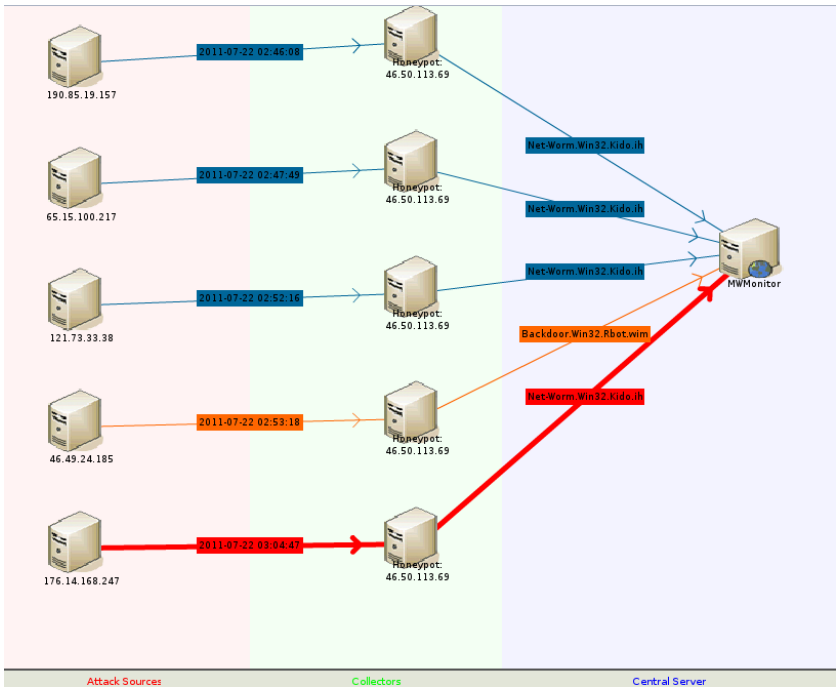


Figura 6.10: *Attack Topology Schema* com os últimos 5 ataques no mês de Julho

Description	Risk
Performs File Modification and Destruction: The executable modifies and destructs files which are not temporary.	
Spawns Processes: The executable produces processes during the execution.	
Performs Registry Activities: The executable reads and modifies registry values. It may also create and monitor registry keys.	

Figura 6.11: Sumário da análise dinâmica do *Net-Worm.Win32.Kido.ih*

Na máquina do utilizador/colaborador:

1. Instalou-se o *Python*.
2. Configurou-se a aplicação *btmclient* no ficheiro *btmclient.ini*.

btmclient.ini

```
1  [mail]
2  smtp=smtp.gmail.com
3  smtport=587
4
5  [binaries]
6  path=./quarantine/
7  \end{itemize}
```

Neste caso, para efeitos de teste do protótipo *mailrobot*, recorreu-se ao serviço de *mail* do Google[4], o *gmail*[5], no entanto existe aqui a liberdade de configuração

[4] http://google.com
[5] https://mail.google.com

Description	Risk
Write to foreign memory areas: This executable tampers with the execution of another process.	
Change Windows Firewall settings: This executable changes some settings of windows firewall.	
Performs File Modification and Destruction: The executable modifies and destructs files which are not temporary.	
Start/Install windows service: This executable starts a windows service. Services have the highest level of privilege in Windows, and are thus useful for a number of malicious purposes.	
Packed Binary: This executable is protected with a packer in order to prevent it from being reverse engineered.	
Autostart capabilities: This executable registers processes to be executed at system start. This could result in unwanted actions to be performed automatically.	
Changes security settings of Internet Explorer: This system alteration could seriously affect safety surfing the World Wide Web.	
Creates files in the Windows system directory: Malware often keeps copies of itself in the Windows directory to stay undetected by users.	
Execution did not terminate correctly: The executable crashed.	
Modify system files: This executable modifies files in the windows system directories.	
Spawns Processes: The executable produces processes during the execution.	
Performs Registry Activities: The executable creates and/or modifies registry entries.	

Figura 6.12: Sumário da análise dinâmica do *Net-Worm.Win32.Kolab.acem*

Description	Risk
Write to foreign memory areas: This executable tampers with the execution of another process.	
Performs File Modification and Destruction: The executable modifies and destructs files which are not temporary.	
Autostart capabilities: This executable registers processes to be executed at system start. This could result in unwanted actions to be performed automatically.	
Changes security settings of Internet Explorer: This system alteration could seriously affect safety surfing the World Wide Web.	
Execution did not terminate correctly: The executable crashed.	
Spawns Processes: The executable produces processes during the execution.	
Performs Registry Activities: The executable creates and/or modifies registry entries.	

Figura 6.13: Sumário da análise dinâmica do *Trojan.Win32.Midgare.ayfl*

do servidor *SMTP* e porta (linhas 2 e 3) correspondente à conta de *mail* utilizador/colaborador. Por fim na linha 6 foi configurada a localização do directório que contém o(s) binário(s) suspeito(s) (a quarentena).

3. Enviou-se o(s) binário(s) suspeito(s) para identificação e análise dinâmica através da aplicação *btmclient*.

btmclient.py

```
1  ############################################################
2  ####    MWMonitor 1.0: MAIL ATTACHMENT SUBMIT CLIENT    ####
3  ############################################################
4  ######  1: LIST QUARANTINE ATTACHMENTS                  ######
5  ######  2: SUBMIT ATTACHMENT                            ######
6  ######  3: QUIT                                         ######
7  ############################################################
8
9
10 Your selection: 1
11
12 f515fbbe5ad60018ca7336597415d547
13 a95ca1b2083f0acdc015a7589d5dadda
14 06d3eba50cb98af21d0a5fbb8828445e
15
16 Your selection: 2
17
18 From: username@gmail.com
```

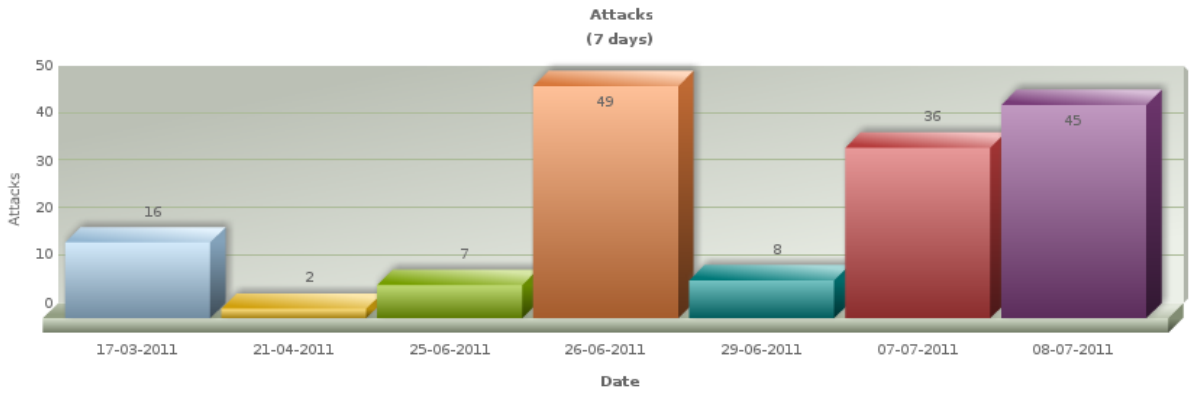


Figura 6.14: Últimos 7 dias de ocorrências de ataque

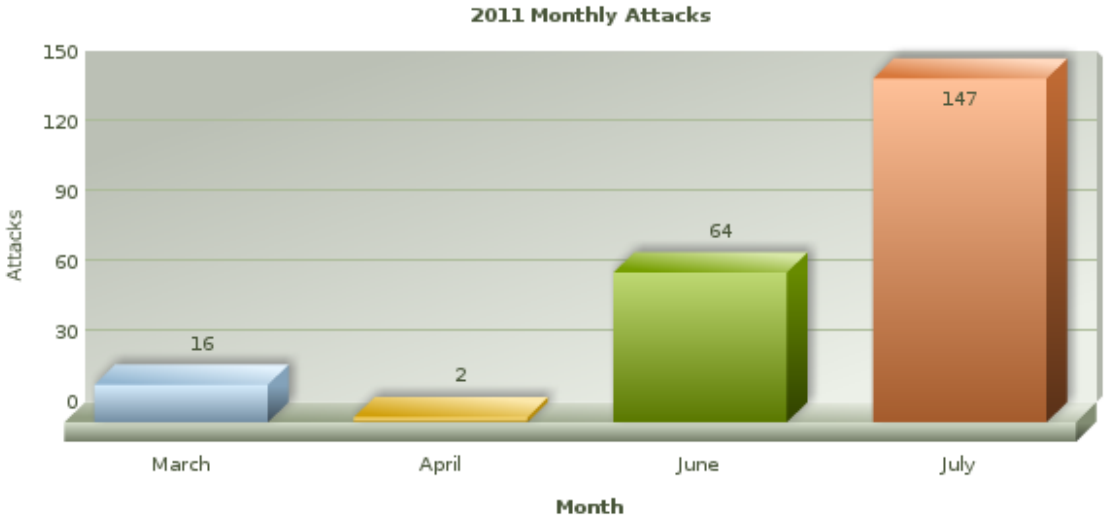


Figura 6.15: Ocorrências de ataque por mês

```
19 Password:
20 Binary: f515fbbe5ad60018ca7336597415d547
21
22 OK
```

Nas linhas 10-14 visualizaram-se os nomes (*MD5*) dos binários suspeitos armazenados na quarentena do utilizador/colaborador, e nas linhas 16-20 introduziu-se o endereço de *e-mail* para envio do binário e respectiva recepção dos resultados de identificação e análise dinâmica (linha 18), a palavra chave associada (linha 19) ao endereço de *e-mail* introduzido na linha 18, o nome do binário que se pretende identificar e analisar na linha 20, e por fim o resultado de envio do binário (OK) na linha 22.

Por fim o binário malicioso foi submetido através do protocolo *SMTP* para uma conta de *mail* especificamente construída para receber binários sus-

peitos de actividade maliciosa (*i.e. wms@gmail.com*).

No servidor central:

1. Configurou-se o componente *mailbinaryrobot* no ficheiro *config.ini*.

/usr/share/mwmonitor/core/config.ini

```
1 [mail]
2 filespath=../mail/
3 imap=imap.gmail.com
4 smtp=smtp.gmail.com
5 smtport=587
6 mailbox=INBOX
7 user=wms@gmail.com
8 passwd=passwd
```

Este componente permite recolher os novos binários suspeitos que foram enviados através da aplicação *btmclient* para uma conta de *mail* (linhas 7 e 8) e está totalmente integrado com o componente *mwmonitor* (e seus subcomponentes) e com a base de dados *attackevents*. Como na actividade experimental foi utilizado o *gmail* com o serviço de *IMAP* activo, o componente *mailbinaryrobot* deve ser configurado para aceder (linhas 2-4) à caixa de entrada (linha 6) da conta de *mail* (linhas 6 e 7) que recebe os binários suspeitos.

2. Configurou-se o *cron service* para execução automática do componente *mailbinaryrobot*.

crontab

```
1 */5 * * * * bash -c "cd /usr/share/mwmonitor/core/;/usr/share/
    mwmonitor/core/mailbinaryrobot.py"
```

Na conta de *mail* do utilizador/colaborador (*username@gmail.com*):

from: wms@gmail.com

```
1 Kaspersky: Net-Worm.Win32.Kolab.tne
2 Symantec: Trojan.FakeAV!gen54
3 McAfee: BackDoor-EYT
4 NOD32: IRC/SdBot
5 Panda: Generic Worm
6 Microsoft: VirTool:Win32/Injector.gen!BF
7 Avast: Win32:Downloader-GIG
8 AVG: Generic21.ETR
9 ClamAV:
10
```

```
11 Anubis: http://anubis.iseclab.org/?action=result&task_id=11
       ef296508949d81471727d59df2392b3
```

Os resultados de identificação por vários motores de anti-*malware* (linhas 1-9) e o *URL* da análise dinâmica (linha 11) do binário suspeito (recorrendo ao *mwmonitor* no servidor central) são enviados para a conta de *mail* do utilizador/colaborador. Como se pode ver na tabela 6.4 foram medidos os tempos de execução do componente *mailbinaryrobot* na sua tarefa de recolha de binários suspeitos da conta de *mail* criada para receber estes binários.

#Binários	Tempo
1	9.839s
3	1m0.083s

Tabela 6.4: Medição do *real execution time* para o componente *mailbinaryrobot*

6.3.4 Cenário 4

No servidor central:

1. Integrar o componente *artifactsrobot* no *mwmonitor*.

   Em baixo pode-se verificar a execução manual do componente *artifactsrobot* para efeitos de teste. Note-se que numa situação normal este componente seria executado pelo *mwmonitor* em conjunto com os sub-componentes *av-submit, ansubmit, alarmwrite, etc*.

   ./artifactsrobot.py –add=0xD289CD91759850640B8C260EDC651D51

```
1  Checking ThreatExpert for file with MD5:
       D289CD91759850640B8C260EDC651D51
2  Analysis exists: http://www.threatexpert.com/report.aspx?md5=
       D289CD91759850640B8C260EDC651D51
3  Added sample with ID 1
4   [FILE] a5bc910a81a30599429f6a72dea004e0 %AppData%\BifroXx\server.
       exe
5   [FILE] a5bc910a81a30599429f6a72dea004e0 %ProgramFiles%\BifroXx\
       server.exe
6   [MUTEX] Bif1234
7   [REGKEY] HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Active Setup\
       Installed Components\{9D71D88C-C598-4935-C5D1-43AA4DB90836}
8   [REGKEY] HKEY_LOCAL_MACHINE\SOFTWARE\BifroXx
9   [REGKEY] HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\
       MediaResources\msvideo
10  [REGKEY] HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\
       MediaResources\msvideo
11  [REGKEY] HKEY_CURRENT_USER\Software\BifroXx
```

Em cima verificaram-se artefactos de infecção como a criação de ficheiros (linhas 4 e 5), chaves de registo (linhas 8-11) e objectos de exclusão mútua (linha 6) criados pelo binário suspeito *d289cd91759850640b8c260edc651d51* (através do serviço *ThreatExpert*). Estes artefactos seriam enviados para cada uma das máquinas críticas que após recepção os guardariam numa base de dados *SQLite* para posterior pesquisa local por artefactos de infecção similares.
Como se pode ver na tabela 6.5 foi ainda medido o *real execution time* para o componente *artifactsrobot*.

#Binários	Tempo
1	0m4.251s

Tabela 6.5: Medição do *real execution time* para o componente *artifactsrobot*

Na(s) máquina(s) crítica(s):

1. Instalar o *Python*, *PyWin32* [6] e o *SQLite*.

2. Configurar o *cron service* para execução automática do componente *artifacts* de *'x'* em *'x'* minutos.
   Execução: *python artifacts.py -frm* (*-f* para pesquisar ficheiros, *-r* para chaves de registo e *-m* para objectos de exclusão mútua).

O objectivo final seria alertar, no caso de existirem artefactos de infecção em máquinas críticas, para uma aplicação *xyz*, equipa operacional de segurança ou administradores de sistemas, de forma a ser corrigido rapidamente o problema.
É ainda importante realçar que a(s) máquina(s) crítica(s) devem ter instalado um sistema operativo *Microsoft Windows XP*, *Vista* ou *7*.

6.4 Resumo

Neste capítulo foi abordada a preparação, experiência e a respectiva avaliação do *WMS* em actividade na rede existente mais propícia a ataques, a *Internet*.

Os resultados foram bastante positivos e esclarecedores, o facto de ser ter tratado de uma actividade experimental numa rede global como a *Internet*, garante quase a *100%* um comportamento de excelência do sistema num ambiente de produção de uma grande rede empresarial.

[6] `http://sourceforge.net/projects/pywin32/`

Capítulo 7

Discussão

7.1 Trabalho preliminar

Inicialmente foi construído um plano realista para a realização deste projecto, foi realizada uma forte investigação na área da segurança forense, nomeadamente através da leitura de capítulos de livros e de diversos artigos científicos, inclusive do projecto *Honeynet*. Após esta fase foi estudada a rede empresarial, o sistema *Pulso* e trabalhos relacionados. Ao longo do estudo dos trabalhos relacionados foram ainda testadas diversas tecnologias utilizadas nesses trabalhos assim como outras alternativas.

7.2 Abordagens alternativas

7.2.1 *dionaea* e *mwcollectd*

Inicialmente uma das principais dificuldades residiu na escolha do *honeypot*, sendo que a dúvida debateu-se sobre a escolha entre dois *honeypots* diferentes.

A investigação iniciou-se sobre o *mwcollectd*, *honeypot* suportado pelos *Kaspersky Labs* e desenvolvido maioritariamente por *Georg Wicherski*. Em paralelo foi testado o *dionaea* (suportado em parte pela *SURFnet* e o *hardware* pelos *Kaspersky Labs*) pela simples razão de se dizer que é o verdadeiro sucessor do bastante conhecido *nepenthes* pois o seu principal programador, *Markus Kötter* já teria estado envolvido no desenvolvimento deste no passado. Foi difícil encontrar diferenças nos dois *honeypots* apenas olhando para as suas funcionalidades principais, mas após uma análise mais profunda e depois de confrontar o *dionaea* com o *mwcollectd*, chegou-se a algumas conclusões que fizeram optar pela utilização do *dionaea* neste projecto. Uma das conclusões que ajudaram na decisão teve a ver com o facto do *dionaea* receber mais contribuição por parte de utilizadores *online* ao nível de melhorias de *performance* no código, correcção de *bugs* e novas funcionalidades, ou seja dispõe de uma grande comunidade de utilizadores *online* que faz

com que exista mais inovação no *dionaea* em relação ao *mwcollectd* que recebe muito poucos *updates*. No entanto verificou-se que o *mwcollectd* permite uma compilação e configuração mais simples.

A conclusão que teve mais influência na escolha do *dionaea* em detrimento do *mwcollectd* teve a ver com a dicotomia rapidez-eficiência dos serviços disponibilizados por ambos. Ou seja, enquanto que o *mwcollectd* se apresentou um pouco mais rápido que o *dionaea* face a tráfego suspeito mais intenso (tráfego esse que felizmente não abunda em redes empresariais), dispõe de menos funcionalidades de captura que o *dionaea*, logo o *mwcollectd* captura menos tráfego supostamente malicioso. O facto do *dionaea* ser mais *feature-rich* e conseguir capturar mais variedade de binários suspeitos de actividade maliciosa do que o seu congénere *mwcollectd*, fez com que a escolha apontasse para o *dionaea* como o *honeypot* do *WMS* por ser o mais eficiente.

7.2.2 *Remote sandbox service* e *Local sandbox service*

Depois de escolhido o *honeypot*, chegou a fase de seleccionar as tecnologias responsáveis pela identificação/detecção e análise dinâmica dos binários suspeitos capturados.

A primeira ideia que surgiu consistiria em ter uma *Local Sandbox* na rede empresarial que passaria pela virtualização (recorrendo a *software* específico para tal, como a *VMWare Workstation* ou a *VirtualBox*[1]) do sistema *Microsoft Windows XP (unpatched)* como *GuestOS* numa máquina multi-processador com *Linux* (*i.e. Debian* ou *RedHat*) como *HostOS*. Essa máquina para realizar a detecção dos binários suspeitos enviados a partir de cada sonda, teria o *ClamAV* instalado localmente.

Para análise dinâmica da execução dos binários suspeitos seria utilizado o *GuestOS* recorrendo de forma automática ao *volatility*[2] para monitorizar as actividades realizadas em memória (fazendo uma análise a respectivos *dumps*), ao *RegShot*[3] para verificar alterações realizadas no *registry*, ao *tshark*[4] ou *tcpdump*[5] para capturar o tráfego de rede gerado e ao *INetSim*[6] para coleccionar os *logs* de rede, antes, durante e após a execução de cada binário.

Como alternativa a esta abordagem ponderou-se ainda a utilização local do serviço de análise dinâmica *Cuckoo Sandbox*[7] (*open-source*).

Estas hipóteses ficaram descartadas pelo simples motivo de aumentarem o

[1] `https://virtualbox.org/`
[2] `https://volatilesystems.com/default/volatility`
[3] `http://sourceforge.net/projects/regshot/`
[4] `http://wireshark.org/docs/man-pages/tshark.html`
[5] `http://www.tcpdump.org/`
[6] `http://inetsim.org/`
[7] `http://cuckoobox.org/`

risco de exposição a actividades maliciosas que possam ocorrer na rede interna, no caso dos binários maliciosos comprometerem o *GuestOS* podem facilmente propagar-se para o *HostOS* e depois para outras máquinas na rede, com grande probabilidade de não serem detectados em tempo útil e poderem gerar uma situação bastante complicada.

Desta forma optou-se por pesquisar por plataformas auxiliares remotas (*Remote Sandbox Service*) de maneira a libertar a rede empresarial de uma árdua tarefa de responsabilidade extrema como é a identificação/detecção e análise dinâmica de binários suspeitos de actividade maliciosa.

Para identificação/detecção de binários foram identificadas várias plataformas remotas de anti-*malware*, como o *VirusTotal*, *NoVirusThanks*[8] e o *Jotti*[9]. Depois de uma análise cuidada de cada umas das plataformas para identificação/detecção de binários construiu-se a tabela 7.1 para sintetizar a comparação entre estas plataformas *free-service*.

Funcionalidade	Jotti	NoVirusThanks	VirusTotal
Número de anti-*malware*	20	24	**42**
Submissão via *web*	X	X	X
Submissão por SSL			**X**
Submissão de URL		X	X
Dispõe de API		X	X
Pesquisa por *File Hash*	X		X
Opção "não guardar ficheiro"		X	
Tamanho máximo do ficheiro	-	20MB	20MB

Tabela 7.1: *VirusTotal*, *NoVirusThanks* e *Jotti*: comparação.

Como se pode constatar na tabela 7.1, a plataforma externa *VirusTotal* apresentou melhores resultados, sendo a solução gratuita mais completa, eficiente e segura. Dados estes resultados foi fácil optar pelo *VirusTotal* como plataforma externa auxiliar de suporte no *WMS* para as tarefas de identificação/detecção de binários suspeitos no *WMS*.

Para análise dinâmica de binários suspeitos de actividade maliciosa foi inicialmente escolhida e usada a plataforma *CWSandbox* só que o facto desta plataforma ter sofrido de problemas na sua infra-estrutura durante meses, não só mostrou alguma falta de robustez como fez com que se procurasse um substituto que desse mais garantias que esta plataforma. A escolha apontou para a plataforma *Anubis* que para além de ser gratuita, baseia-se em vários artigos científicos premiados. Para além disso é uma plataforma bastante robusta, rápida e eficiente tanto no

[8] `http://vscan.novirusthanks.org/`
[9] `http://virusscan.jotti.org/`

processo de análise dinâmica como no retorno dos resultados. O facto de se optar por serviços de identificação e análise dinâmica remota obrigou a uma alteração na arquitectura do *WMS*. Basicamente no início assumiu-se erroneamente uma arquitectura centralizada que dispunha de sondas com ligação à *Internet*, o que numa rede empresarial tipicamente não correspondente à realidade, estas apenas têm conectividade na *Intranet*, ou seja só capturam tráfego gerado na rede interna.

Sabia-se também à priori que as sondas poderiam não ter recursos computacionais suficientes para suportar todo o *back-end* do *WMS*. Visto tal facto, optou-se por formar uma federação de sondas (uma *honeynet*) que comunica cooperativamente com um servidor central numa arquitectura distribuída, é o servidor central que dispõe de ligação à *Internet* assim como mais recursos computacionais disponíveis para rápido acesso às plataformas externas auxiliares (*VirusTotal* e *Anubis*).

7.3 Funcionamento interno

O facto de se ter optado pelo *dionaea honeypot* fez com que ganhasse mais experiência na área de *reverse engineering*. Desenvolvi várias *skills* sobre ferramentas e técnicas para combate ao *malware* e estudei bastante os tipos de *malware* que envolvem propagação automática (*worms* e *bots*).

Com a escolha do *dionaea honeypot* acabei por contribuir com a descoberta de *bugs* e no fundo até corrigir alguns. Notou-se desde muito cedo que havia uma enorme despreocupação em relação a alguns componentes do *honeypot* nomeadamente ao *submithttp*, a razão deve-se à aposta na utilização do protocolo *XMPP* com o *dionaea*, no entanto numa grande rede empresarial a integração de um novo protocolo deve ser muito bem pensada e no fundo a utilização do protocolo *XMPP* poderia ser uma grande vantagem em certos aspectos mas uma grande desvantagem noutros.

Um ponto que deve ficar bastante claro é que o sistema permite apenas a captura de *malware* que inclui um mecanismo de propagação automática (*worms* e *bots*), mesmo sabendo que o componente de *back-end mwmonitor* e os seus subordinados *avsubmit* e *ansubmit* permitem identificar e analisar dinamicamente "todo" o tipo de *malware*. Neste caso pode-se mesmo dizer que o *malware* identificado e analisado através do *mwmonitor* no servidor central está fortemente relacionado e dependente da categoria de *malware* que neste momento um *honeypot* (neste caso o *dionaea*) consegue capturar, daí o foco do projecto na captura de *worms* e *bots*. O protótipo *mailrobot* permite contornar esta barreira, pois ao contrário da utilização principal do sistema (com o *honeypot*) que funciona em modo automático, o *mail-*

robot funciona parcialmente no modo manual (*btmclient*).

No que diz respeito à implementação do projecto, ficou em falta o desenvolvimento do cliente e servidor para o protótipo *artifactsrobot* que permitiriam o envio dos artefactos de infecção para as máquinas críticas assim como o servidor que estaria em cada uma dessas máquinas para receber os artefactos.

7.4 Recursos

Pode-se dizer que desde início houve uma especial preocupação com o consumo de recursos computacionais no que diz respeito às tecnologias utilizadas.

O sistema foi optimizado para garantir um rápido, eficaz e correcto funcionamento mesmo que se tratem de máquinas com recursos computacionais reduzidos.

Na PTC os recursos computacionais disponíveis foram mais que suficientes para a realização completa do projecto, no entanto para a realização da actividade experimental houve necessidade de ligar a(s) sonda(s) à *Internet*, pois durante o tempo que o sistema esteve em execução na rede interna da PT, não surgiu qualquer ocorrência de ataque através de propagação de *malware*.

7.5 Resultados

É importante realçar o facto de ter sido cumprido o melhor possível o plano construído inicialmente para o desenvolvimento deste projecto. Pode-se afirmar de uma forma muito realista que não houve qualquer desvio negativo no que diz respeito ao plano de trabalho inicialmente estipulado, aliás pode-se dizer que houve um desvio positivo, pois foram realizadas aplicações extra (como o *mail-robot* e o *artifactsrobot*) que inicialmente não estavam planeadas.

O tempo é em quase tudo o maior aliado, e aqui não é excepção, no caso da actividade experimental a exposição dos *honeypot(s)* durante mais tempo na rede empresarial poderia ter colhido os seus frutos, no entanto existe a outra face da moeda em que poderia no final não ter dados suficientes (ou nem existirem) para avaliar experimentalmente o sistema construído.

Capítulo 8

Conclusão

Conclui-se muito realisticamente que os resultados foram bastante positivos e que a solução implementada pode ser uma mais valia numa grande rede empresarial. Para tirar o melhor partido da solução numa rede empresarial pode-se integrar os serviços de *QoP* garantidos por esta solução com os de *QoS* e *QoM* garantidos tipicamente por outras soluções.
Verificou-se que tanto o *dionaea honeypot* como o *back-end mwmonitor* e as plataformas remotas *VirusTotal* e *Anubis* fornecem uma solução bastante eficaz e aplicável em ambientes de produção com um elevado volume de tráfego suspeito de actividade maliciosa. Neste projecto conseguiu-se ainda perceber as exigências em termos de segurança da informação impostas pelas grandes empresas, pois sabe-se de antemão que estas empresas hoje em dia têm de adoptar bons mecanismos de gestão e controlo dos sistemas na rede empresarial para que o nível de produtividade não seja afectado. Só desta forma uma empresa consegue melhorar gradualmente todo o processo de negócio e os respectivos índices de produtividade face à forte concorrência.

O estágio na PTC deu para perceber bem estas exigências, assim como deu para entender que os mecanismos de protecção de informação, por forma a garantir a segurança da rede empresarial, devem funcionar da forma mais passiva/não intrusiva possível.
Esta solução dá uma excelente resposta no que diz respeito à necessidade de uma grande empresa proteger a sua rede interna tanto dos utilizadores/colaboradores internos e autorizados, como dos utilizadores externos e não autorizados (no caso de haver ligação a uma rede externa como a *Internet*).

Em suma, pode-se afirmar que os objectivos principais e extra (Capítulo 1.2) foram cumpridos com sucesso. Conseguiu-se implementar todo o projecto recorrendo por um lado a tecnologia de terceiros *free* e/ou *open-source* (capítulo 4) e por outro lado, tanto o *submithttp* para o *dionaea honeypot* como o *receivehttp* e o *back-end mwmonitor* (*avsubmit*, *ansubmit*, *alarm*, *alarmwrite*, etc) foram programados e

bem documentados com vista a futuras extensões (*i.e. mailrobot* e *artifactsrobot* usam o *mwmonitor*).
O *WMSi* foi implementado com o mesmo tipo de regras, ou seja, com o objectivo de ser facilmente estendido com novas funcionalidades.

No que diz respeito aos resultados obtidos pode-se afirmar que foram bastante interessantes tendo em conta o tempo em que o sistema esteve em actividade e avaliação.

O balanço geral do projecto foi em todas as etapas de desenvolvimento bastante positivo, cada etapa foi uma mais-valia em termos de especialização nesta área da segurança informática que é tão fascinante, apaixonante, importantíssima e infelizmente tantas vezes esquecida.

8.1 Divulgação

O estágio na equipa de DES da PTC foi uma mais valia não só pela oportunidade em si como pela divulgação deste projecto. Já numa fase avançada do estágio o projecto foi apresentado[1] no prestigiante seminário *Cyber Security: An action-based approach* que decorreu no Auditório da Marinha em Lisboa, e foi organizado pela *National Security Cabinet* (GNS)[2], *EuroDefense*-Portugal[3] e AFCEA-Portugal (Associação para as Comunicações, Electrónica, Informações e Sistemas de Informação para Profissionais)[4]. Este seminário contou com a participação das principais entidades nacionais relevantes no âmbito desta temática, na vertente militar, governamental, académica e industrial.
Não menos importante, o projecto foi ainda apresentado na visita da Direcção de Sistemas de Informação (DSI) da EDP à PT.

8.2 Trabalho futuro

Depois do desenvolvimento do projecto surgiram algumas ideias para trabalho futuro e consequente evolução do *WMS*:

- **mecanismo de *fuzzy hashes***
  As *fuzzy hashes* podem ser um mecanismo poderoso para determinar rigorosamente a similaridade entre ficheiros. Para uma variante de um binário malicioso o resultado da sua identificação a partir do seu *MD5* terá muito provavelmente o mesmo resultado que o binário malicioso original. Este

[1] `http://www.ptempresas.pt/Noticias/Pages/2011_05_16.aspx`
[2] `http://www.gns.gov.pt/gns/pt/`
[3] `http://eurodefense.aip.pt/`
[4] `http://www.afceaportugal.pt/`

facto deve-se a mecanismos (muitas vezes erróneos) de *clustering* das assinaturas de *malware* implementados nas várias plataformas externas que permitem identificar e analisar binários suspeitos, devolvendo inevitavelmente o mesmo resultado de identificação para dois binários que na realidade são um pouco diferentes, logo podem não ter o mesmo comportamento. Como os vários componentes do *WMS* foram maioritariamente escritos em *Python*, através do módulo *pyssdeep*[5], que permite utilizar as funcionalidades do *ssdeep*[6], poder-se-ia gerar *fuzzy hashes* para binários maliciosos já identificados, e determinar em termos de percentagem a similaridade entre eles. Esta seria uma excelente forma de desambiguar os mesmos resultados de identificação para dois binários maliciosos que na realidade são diferentes (original e variante). Adicionalmente com este mecanismo podem-se realizar outras tarefas adicionais bastante interessantes como por exemplo detectar *hollow processes* em máquinas críticas mesmo que não existam artefactos (alterações no *registry*, etc) de infecção para realizar pesquisas nessas máquinas. Um *hollow process* pode ser um programa legítimo (como o *notepad.exe*) que é iniciado por *malware*. Uma vez que o programa esteja em execução o *malware* em causa substitui-lhe as instruções legítimas com instruções maliciosas. Neste caso o *ssdeep* (através das *ssdeep hashes*) detectaria a injecção de código porque o ficheiro *notepad.exe* no disco seria significativamente diferente do que estaria em *RAM*. Tipicamente dois binários são diferentes quando a percentagem de similaridade entre duas *fuzzy hashes* são inferiores a *75-80%*. Em suma, este é um mecanismo bastante poderoso que poderia muito bem ser integrado no *WMS* dando-lhe certamente um valor acrescentado no que diz respeito a ambiguidades obtidas nos resultados de identificação de binários maliciosos.

- ***Attack Topology Schema***
  O método de visualização *Attack Topology Schema* tem grande margem de crescimento numa futura extensão do *front-end*, pois permitirá ter uma única *wallboard* com tanto (ou mais) poder visual e interpretativo como os vários *dashboards* construídos e integrados no *WMSi* para efeitos de monitorização de segurança.

- **integração no sistema *Pulso***
  Seria bastante interessante a integração do *WMS*, para efeitos de alarmística, num portal de monitorização de risco técnico como o *Pulso*.
  Dotar a plataforma *Pulso* da capacidade de automaticamente identificar/de-

[5] `http://code.google.com/p/pyssdeep`
[6] `http://ssdeep.sourceforge.net`

tectar, analisar e monitorizar ocorrências de *malware* propagado automaticamente, neste caso, pela rede interna da PT através da integração do *WMS*, seria certamente uma mais valia competitiva face a outros sistemas concorrentes.

- **melhoria/conclusão do protótipo *artifactsrobot***
  Como já foi mencionado no capítulo anterior, para a conclusão do *artifactsrobot*, faltará apenas implementar o cliente que simplesmente estará responsável por enviar os artefactos de infecção para cada uma das máquinas críticas e o servidor que estará responsável por receber e armazenar localmente esses artefactos que depois serão utilizados pelo componente *artifacts* que fará a pesquisa local.

- ***honeynets* cooperativas**
  Um dos módulos mais interessantes e inovadores que vem com o *dionaea honeypot* é o *XMPP* (é necessária a activação do *logxmpp* na secção *ihandlers* do *dionaea.conf*), que permite comunicações em tempo real para partilha de eventos de ataque (notificações) e respectivos binários suspeitos de actividade maliciosa.
  Essa partilha é realizada entre clientes (*dionaea honeypot*) através de um servidor *XMPP* dedicado (prosody[7] ou ejabberd[8]) que mantém o canal de partilha de dados entre os clientes (semelhante a uma sala de *chat*).
  Esta tecnologia forneceria um ambiente distribuído e cooperativo altamente poderoso no caso de existirem relações com outras grandes empresas que também tenham em actividade o *dionaea honeypot* com este módulo activo.
  Desta forma haveriam várias *honeynets* cooperativas dependendo do número de empresas envolvidas, todas iriam beneficiar ao longo do tempo com este mecanismo, melhorando desta forma a segurança das suas redes internas. Esta solução como quase todas as soluções tem um senão, este senão tem a ver com o facto desta solução basear-se na confiança às cegas entre *honeypots/honeynets* oriundas de diferentes entidades, exigir tipicamente uma máquina servidora dedicada, e aumentar (dependendo do número de clientes) o volume de tráfego.

- **inferência do caminho de propagação do *worm* ou *bot* capturado**
  A idéia subjacente iria basear-se no estudo e posterior utilização do algoritmo de *Monte Carlo EM (MCEM)*[35] para inferir o caminho de propagação de um *worm* ou *bot* numa grande rede empresarial a partir de dados incompletos obtidos por cada *honeypot*.

[7] `http://prosody.im`
[8] `http://ejabberd.im`

Bibliografia

[1] José A. S. Alegria, Tiago F. R. Carvalho, and Ricardo G. Ramalho. *Uma experiência open source para "tomar o pulso" e "ter pulso" sobre a função sistemas e tecnologias de informação*.

[2] Ulrich Bayer. *Master's Thesis - TTAnalyze: A Tool for Analyzing Malware*.

[3] Ulrich Bayer, Imam Habibi, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. *Insights into current malware behavior*. 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), Boston, 2009.

[4] Ulrich Bayer, Christopher Kruegel, and Engin Kirda. *TTAnalyze: A tool for analyzing malware*. 15th European Institute for Computer Antivirus Research Annual Conference, 2006.

[5] Ulrich Bayer, Paolo Milani, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. *Behavior-based malware clustering*. 16th Annual Network and Distributed System Security Symposium, San Diego, 2009.

[6] Ulrich Bayer, Andreas Moser, Christopher Kruegel, and Engin Kirda. *Dynamic analysis of malicious code*. Journal in Computer Virology, Springer Computer Science.

[7] Bill Blunden. *The Rootkit Arsenal: escape and evasion in the dark corners of the system*. Jones & Bartlett Publishers, 1st edition, 2009.

[8] Evan Broder. *Transparent detection of computer malware using virtualization*. MIT.

[9] Alberto Bruno. *Security Metrics to Evaluate Quality of Protection*. 2011.

[10] Miguel Correia and Paulo Sousa. *Segurança no Software*. FCA, 1st edition, 2010.

[11] Nicolas Falliere, Liam O Murchu, and Eric Chien. *W32.Stuxnet Dossier*. Symantec Security Response, 2011.

[12] Noah Gift and Jeremy Jones. *Python for Unix and Linux System Administration*. O'Reilly, 2008.

[13] Allen Harper, Shon Harris, Jonathan Ness, Chris Eagle, Gideon Lenkey, and Terron Williams. *Gray Hat Hacking The Ethical Hackers Handbook*. McGraw-Hill, 3rd edition, 2011.

[14] Lance Hayden. *IT Security Metrics - A Practical Framework for Measuring Security Protecting Data*. McGraw-Hill, 2010.

[15] Thorsten Holz and Frederic Raynal. *Detecting Honeypots and other suspicious environments*. United States Military Academy, West Point, NY, 2005.

[16] Honeynet. *Know Your Enemy Papers*.

[17] John Hoopes. *Virtualization for Security: Including Sandboxing, Disaster Recovery, High Availability, Forensic Analysis, and Honeypotting*. Syngress, 2008.

[18] Nwokedi Idika and Aditya P. Mathur. *A Survey of malware detection techniques*. Purdue University, 2007.

[19] Harleen Juneja. *Detecting worms using honeypots*. ME Punjab Engineering College.

[20] David Kennedy, Jim O'Gorman, and Devon Kearns. *Metasploit: The Penetration Tester's Guide*. No Starch Press, 1st edition, 2011.

[21] Jack Koziol and Chris Anley. *The Shellcoder's Handbook: Discovering and Exploiting Security Holes*. Wiley, 2nd edition, 2007.

[22] James Kurose and Keith Ross. *Computer Networking: A Top-Down Approach*. Addison-Wesley, 5th edition, 2009.

[23] Mark Lutz. *Learning Python*. O'Reilly, 3rd edition, 2007.

[24] Cameron H. Malin and James M. Aquilina. *Malware forensics: investigating and analyzing malicious code*. Syngress, 2008.

[25] Cameron H. Malin and James M. Aquilina. *Malware Forensics: Investigating and Analyzing Malicious Code*. Syngress, 1st edition, 2008.

[26] Robert McGrew, B Rayford, Vaughn, and JR. *Experiences with honeypot systems: development, deployment, and analysis*. PhD Mississippi State University, 2011.

[27] John Mitchell. *Network Worms and Bots*. 2008.

[28] Evi Nemeth, Garth Snyder, Trent R. Hein, and Ben Whaley. *Unix and Linux System Administration Handbook*. Prentice Hall, 4th edition, 2011.

[29] Nuno Neves. *Honeypots and virtualization*. FCUL-CMU-MSI, 2008.

[30] Nuno Neves. *Security Metrics*. FCUL-CMU-MSI, 2008.

[31] Vinod P., V.Laxmi, and M.S.Gaur. *Survey on Malware Detection Methods*.

[32] Georgios Portokalidi. *Zero-hour worm detection and containment using honeypots*. 2005.

[33] Niels Provos and Thorsten Holz. *Virtual Honeypots: from botnet tracking to intrusion detection*. Addison-Wesley Professional, 1st edition, 2007.

[34] Xinzhou Qin and David Dagon. *Worm detection using local network*. Georgia Institute of Technology.

[35] Michael G. Rabbat, Mário A. T. Figueiredo, and Robert D. Nowak. *Network Inference from Co-Occurrences*.

[36] Bruce Schneier. *Applied Cryptography*. Wiley, 1996.

[37] Donn Seeley. *A Tour of the Worm*. University of Utah, 2005.

[38] Ed Skoudis and Zeltser Lenny. *Malware: fighting malicious code*. Prentice Hall, 2003.

[39] Lance Spitzner. *Honeypots: tracking hackers*. Addison-Wesley Professional, 2002.

[40] William Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 5th edition, 2010.

[41] Peter Szor. *The art of computer virus research and defense*. 2005.

[42] Paulo Veríssimo and Luís Rodrigues. *Distributed Systems For System Architects*. Kluwer Academic Publishers, 2001.

[43] Jung Hsiang Wang. *The nepenthes platform: an efficient approach to collect malware*. Cryptography and Network Security Laboratory, 2008.

[44] Nicholas Weaver, Vern Paxson, Stuart Stanifor, and Robert Cunninghan. *A taxonomy of computer worms*.

[45] Jianhong Xia, Jiang Wu, Sarma Vangala, and Lixin Gao. *Effective worm detection for various scan techniques*. University of Massachusetts at Amherst.